Journal of New Music Research

ISSN: 0929-8215 (Print) 1744-5027 (Online) Journal homepage: https://www.tandfonline.com/loi/nnmr20

The CrossSong Puzzle: Developing a Logic Puzzle for Musical Thinking

Jordan B.L. Smith, Jun Kato, Satoru Fukayama, Graham Percival & Masataka Goto

To cite this article: Jordan B.L. Smith, Jun Kato, Satoru Fukayama, Graham Percival & Masataka Goto (2017) The CrossSong Puzzle: Developing a Logic Puzzle for Musical Thinking, Journal of New Music Research, 46:3, 213-228, DOI: 10.1080/09298215.2017.1303519

To link to this article: https://doi.org/10.1080/09298215.2017.1303519

Published online: 21 Mar 2017.

Journal of New Music Research, 2017
Vol. 46, No. 3, 213–228, https://doi.org/10.1080/09298215.2017.1303519

Routledge
Taylor & Francis Group

The CrossSong Puzzle: Developing a Logic Puzzle for Musical Thinking*

Jordan B.L. Smith and Jun Kato, Satoru Fukayama, Graham Percival and Masataka Goto

National Institute of Advanced Industrial Science and Technology (AIST), Japan

(Received 27 July 2016; accepted 23 February 2017)

Abstract

There is considerable interest in music-based games and apps. However, in existing games, music generally serves as an accompaniment or as a reward for progress. We set out to design a game where paying attention to the music would be essential to making deductions and solving the puzzle. The result is the CrossSong Puzzle, a novel type of music-based logic puzzle that integrates musical and logical reasoning. The game presents a player with a grid of tiles, each representing a mash-up of excerpts from two different songs. The goal is to rearrange the tiles so that each row and column plays a continuous musical excerpt. To create puzzles, we implemented an algorithm to automatically identify a set of song fragments to fill a grid such that each tile contains an acceptable mash-up. We present several optimisations to speed up the search for high-quality grids. We also discuss the iterative design of the game's interface and present the results of a user evaluation of the final design. Finally, we present some insights learned from the experience which we believe are important to developing music-based puzzle games that are entertaining, feasible and that challenge one's ability to think about music.

Keywords: games, puzzles, mash-ups, interfaces, software

1. Introduction

Why is listening to music enjoyable? One hypothesis is that a listener's pleasure derives (at least in part) from one's ability to detect patterns in the music, thereby 'compressing' it in one's mind (Schmidhuber, 2009). There is some evidence that, compared to other works, compositions widely regarded as musical masterpieces may be more compressible, despite having a more complex surface representation (Hudson, 2011). If this hypothesis is correct, then pattern identification is central to the enjoyment of both music and puzzles. In logic puzzles, such as sudoku, solvers are tasked with identifying patterns and discovering strategies to take advantage of them in order to complete the puzzle. To solve a puzzle can be seen as an act of entropy reduction: to assemble a scattered pile of jigsaw pieces into an image; to fill a series of blanks, whose values are uncertain, with fixed values.

If music and puzzles are enjoyable for similar reasons, is it possible to devise an activity that combines the two? Despite some historical precedents (discussed in Section 2), few activities target those with an interest in both. This may be due to an inherent difference that puts the pastimes at odds with each other: puzzles have no tempo or schedule—they are solved at the solver's own pace—while music is defined by its happening in time, and interruptions or sudden changes in rhythm or playback will detract from the experience. Devising a satisfying combination of active listening and puzzle-solving is therefore a difficult challenge.

In this work, we have embraced this challenge. The result is the CrossSong Puzzle (see Figure 1), a real-time puzzle game in which the solver listens to the audio 'clues' without interruptions. In the 4×4 grid of tiles, each row and column represents a four-part excerpt of a song. Each tile thus represents a mash-up of two songs. The solver is presented with a scrambled grid, and the object of the puzzle is to reconstruct the excerpts by listening to the tiles and rearranging them. The excerpts are metrically aligned and time-stretched to the same duration so that all beats coincide. Gameplay is continuous: the tiles play one after the other, smoothing the discontinuities between the different mashups. The result is akin to an indefinite, reconfigurable remix of the source songs.

Correspondence: Jordan B.L. Smith, National Institute of Advanced Industrial Science and Technology (AIST), Central 2, 1-1-1 Umezono; Tsukuba, Ibaraki; 305-8568; Japan. Email: jordan.smith@aist.go.jp

*An early version of this article won the Best Paper Award at the Sound and Music Computing conference in Maynooth, Ireland, in 2015.

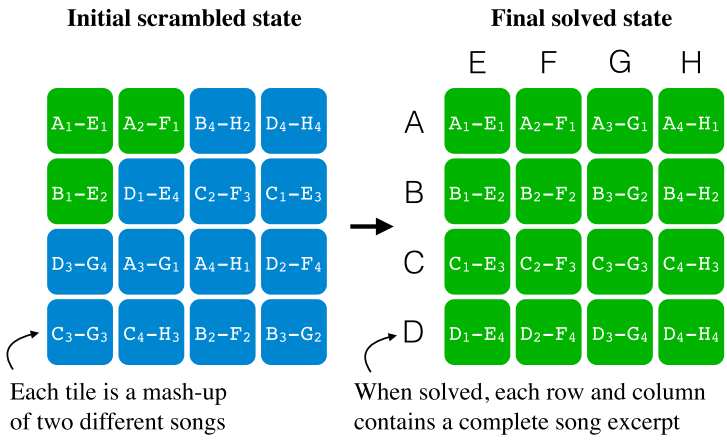


Fig. 1. CrossSong puzzle overview. Green tiles indicate correct placement. The solver cannot see the labels and must deduce the correct order by listening to the tiles. Demo application and gameplay video available at: https://staff.aist.go.jp/jun.kato/CrossSong/.

The CrossSong puzzle may call to mind the classic 4×4 sliding-tile puzzle, in which the goal is to reconstruct an image given similar constraints. However, its highly constrained construction more closely resembles that of a crossword puzzle. A cruciverbalist (a constructor of crosswords) must find suitable words to fill a grid such that wherever two words cross, the letters match. Likewise, to make a pleasing CrossSong puzzle, we must find suitable song excerpts such that wherever two songs cross, a pleasing mash-up is made. Discovering a set of excerpts where this is possible is challenging but, we anticipate, essential for the enjoyment of the puzzle. The algorithm we developed for doing this, based on the work of Davies, Hamel, Yoshii, and Goto (2014), is one of our main contributions.

What is the purpose of our game? Besides being fun on its own, a music-based puzzle game may prove useful for music education. Common sense, partly backed up by a body of literature, argues that games and puzzles are useful for exercising basic cognitive skills (Griffiths, 2002). Even advanced subject knowledge, like molecular chemistry, can be reinforced with the appropriate use of games (Crute, 2010). Unique among music-related games (discussed in detail in the next section), CrossSong has been designed to encourage 'musical thinking.' The puzzle requires solvers to isolate rhythms, timbres and melodies in their mind in order to identify connections between the tiles. Such careful listening may flex their musicianship.

Section 2 reviews existing combinations of music and puzzles, as well as previous work in mashability estimation. The rest of the article is structured around the main contributions of this work: Section 3 gives a formal overview of the proposed CrossSong Puzzle design, including gameplay and implementation; Section 4 describes the grid construction algorithm and a test of several grid search heuristics; Section 5 discusses the iterative development and testing of the interface, and the design principles which guided it; Section 6 presents the results of an evaluation of the finalised design; and we discuss the project and give concluding remarks in Sections 7 and 8.

2. Related work

In this section, we first review prior examples of games based on music. Second, we review existing software for creating and estimating the quality of mash-ups.

2.1 Musical puzzles and games

The earliest type of musical puzzle may be the puzzle canon, which has existed since at least the fourteenth century (Ciconia, 1993, p. 411). In a canon, a melody is set against a delayed, possibly transposed repetition of itself. In a puzzle canon, a single melody is given, and the solver must determine the delay and transposition required so that the result obeys the rules of counterpoint. After several dissonant attempts, the solver is rewarded with the harmonious result once they find the correct solution. It is possible to propose any melody as a puzzle canon, without a solution in mind—such is the origin of Bach's famous 'Musical Offering'—but in this case a solution is not guaranteed to exist.

Puzzle canons may be treated as a curiosity, but in truth, most exercises in counterpoint training can also be seen as puzzles. Species counterpoint, a practice that is a fundamental part of many undergraduate music theory curricula, is a practice of writing music with strict melodic and harmonic constraints. Typical counterpoint exercises, like writing a melody above a bassline or harmonising a melody as a chorale, can be regarded as logic puzzles. Although there are usually many solutions, finding any solution without breaking a rule is challenging enough.

An example of turning music-making into a game, rather than a puzzle, is the musical dice game, a form popular in Western Europe in the 1700s (Hedges, 1978), in which random rolls of the dice were used to choose a selection of score fragments which could then be performed for one's amusement. The dice game is an important ancestor to modern algorithmic composition (Nierhaus, 2009, p. 36). A recent system for generating medleys of existing recordings is also inspired by the dice game (Lin, Liu, Jang, & Wu, 2015).

Today, there are many web- and smartphone-based games which are based on music; however, a partial survey (Dubus, Hansen, & Bresin, 2012) suggested that the market is dominated by sound banks, multimedia players, instrument emulators and music-creation apps like synthesisers and sequencers. In those apps and games we are aware of that do combine music and puzzles, the link between the music and the puzzle mechanics is rarely strong. In most cases, the logical reasoning is separate from the music, which instead serves as a progress indicator or as a kind of reward, generated by the game state or triggered by the correct solution to the puzzle.

For example, in the puzzle game *Auditorium*,[1] players must arrange attractors, repellers and other items in order to guide a stream of light particles to a set of sinks. Each sink activates part of the soundtrack, so the music reflects one's progress in the level. The gradual superposition of musical parts makes a pleasing soundtrack, but the game could be played just as easily with no audio: all of the musical cues that indicate progress are redundantly matched with visual cues about which sinks are active. The same observation—that the music does not convey any crucial information—could be made for other music puzzle games such as *Chime*,[2] *Lumines*,[3] and *FRACT OSC*.[4]

A related category of music-based video games, known as rhythm games, includes *DrumMania* and the hugely popular *Guitar Hero*[5] series. In these games, the challenge is to execute a sequence of physical actions indicated by a stream of visual cues. In contrast to the aforementioned music puzzle games, hearing the music is essential for rhythm games in order to entrain to the beat. Also, since the cues often match the melodic contour of the music, musical knowledge could help a player entrain to the cues. However, these are better described as physical challenges than as puzzles.

What is missing is a puzzle where the music is the *source* of information used by the solver, and which demands careful listening. To our knowledge, the only predecessor with this feature is the prototype game developed by Hansen, Hiraga, Li, and Wang (2013), who developed a musical analogue of a jigsaw puzzle. A 15-second excerpt of music is divided into pieces and the solver's goal is to arrange the pieces from left to right in order to reconstruct the original excerpt. As an added challenge, the audio of several pieces has been randomly transposed; the solver must detect and undo these transpositions in order to complete the puzzle.

Their design has limitations that ours aims to overcome. First, playback is not continuous: puzzle pieces must be individually triggered, which reduces the immersiveness of the gameplay and of the music. Second, each musical excerpt is divided into pieces at arbitrary timepoints, so the resulting pieces do not sound like coherent fragments. Thus, when the pieces are in incorrect order, the result can sound not only incorrect but also unmusical. It would be preferable to divide the fragments only at beat or downbeat positions. In fact, some music psychology experiments suggest that music that is divided and reassembled at a sensible timescale can be as enjoyable to listeners as the original (Upham & Farbood, 2013).

2.2 Automatic remixing and level creation

We propose a new type of puzzle, the CrossSong puzzle (described in the next section), but in doing so we must also propose a method of creating instances of this puzzle. Puzzles could be created by hand, but this would be cumbersome, and would preclude the possibility of generating puzzles on-demand for users who want puzzles that feature music of their choosing. Thus we would prefer a fully-automated grid-creation algorithm. Some modern mobile games use automated music analysis to generate level content, including *AudioSurf*[6] and *Record Run*.[7] Levels in *Guitar Hero* are created individually by humans, but the research prototype *Beat the Beat* (Jordan et al., 2012) uses rhythmic information extracted from the audio to generate levels for a *Guitar Hero*-like minigame.

For the CrossSong puzzle, we require an algorithm that can do two things: first, automatically align the beat of two pieces with beat-tracking; and second, estimate the quality of the resulting mash-up at multiple shifts in pitch. The problem of beat tracking constitutes a field of its own; see Hainsworth (2006), among others, for a review. Downbeat tracking is an even harder problem that continues to inspire a large amount of research (see, e.g. Krebs, Böck, & Widmer, 2013). Many tools are capable of estimating beat locations, and some of these directly facilitate the creation of mash-ups, such as the Echo Nest Remix API[8] (discontinued as of May 2016, and superceded by the open-source toolbox Amen[9]).

A growing number of systems can use this information to create mash-ups autonomously. Beat-Sync-Mash-Coder (Griffin, Kim, & Turnbull, 2010) computes beat information and uses it to automatically synchronise and loop the playback of two excerpts, but the excerpts must be extracted from full-length audio files manually by the user, and the system does not attempt to match the pitch of the excerpts. The commercial system Mixed In Key[10] estimates the mutual harmonic compatibility of all songs in a collection, and can recommend source material for users to create mash-ups on their own. However, the compatibility estimate is on a song-to-song basis with no timing information; this is too coarse for our purpose, since the compatibility of two excerpts can be greatly affected by the phase of the excerpts. The dice-game-inspired system

[1] http://www.cipherprime.com/games/auditorium/.
[2] http://www.chimegame.com/.
[3] http://lumines.jp/.
[4] http://fractgame.com/.
[5] http://www.guitarhero.com/.
[6] http://www.audio-surf.com/.
[7] http://www.harmonixmusic.com/games/record-run/.
[8] http://echonest.github.io/remix/.
[9] https://github.com/algorithmic-music-exploration/amen.
[10] http://mashup.mixedinkey.com/HowTo.

cited earlier (Lin et al., 2015) produces pleasing medleys, but these are concatenations rather than overlapping mash-ups of music.

Among existing systems, AutoMashUpper (Davies et al., 2014) fulfils our requirements best. First, it performs beat, downbeat and phrase-level boundary detection; this is important because mash-ups between phrases that are intact and aligned downbeat-to-downbeat sound better, at least in the opinion of the authors. Second, it estimates the harmonic, rhythmic and spectral compatibility of two phrases at all possible shifts in pitch and time. The harmonic compatibility of two segments is taken as the correlation between chromagrams estimated from the audio. Rhythmic compatibility is estimated in the same way, using a rhythmic feature derived from the pattern of estimated kick and snare onsets. Finally, the coarse spectra from each segment are compared; the flatter their sum, the more the two excerpts are deemed to have complementary spectra, and the greater their mashability. Details of this algorithm can be found in Davies et al. (2014). A competing algorithm, proposed in Lee, Lin, Yao, Lee, and Wu (2015), uses a similar approach but also attempts to ensure fluid transitions between successive sections of the remix, which is not a concern for the present application. In Section 4, we describe how AutoMashUpper was adapted for our needs.

It should be noted that AutoMashUpper makes many strong assumptions about the rhythmic regularity of the piece: constant tempo, constant 4/4 metre, and for the most part, phrases that are 2^n bars long. These assumptions clearly do not apply to all music, so they present a problem for our system. The user should be aware of this constraint and avoid selecting music in different time signatures. In the future, an automatic metre-detection step could be developed to quickly warn users of incompatible songs.

3. CrossSong puzzle

The CrossSong Puzzle was described briefly in the introduction. In this section, we explain the design and construction of the puzzle in more detail. Later in Section 5, we explain how our design evolved over a series of user tests.

In its solved state, the puzzle contains excerpts from 8 different songs, labelled A–H, one for each row and column of the grid. (See Figure 2.) Each song excerpt has the same length in beats, which is some multiple of four, so that each may be divided into four parts: X_1–X_4 for song X. With each excerpt time-stretched to the same duration and spread across four tiles, each tile is assigned two parts that play simultaneously as a mash-up: one from the 'across' song and one from the 'down' song.

The solver begins the puzzle with the tiles arranged randomly, and their task is to determine the correct order by listening to the tiles. Due to the symmetry of the puzzle, there exist two solutions (i.e. the arrangement in Figure 2 and its transpose); therefore, we must fix one off-diagonal tile in place and reveal its correctness to the solver.

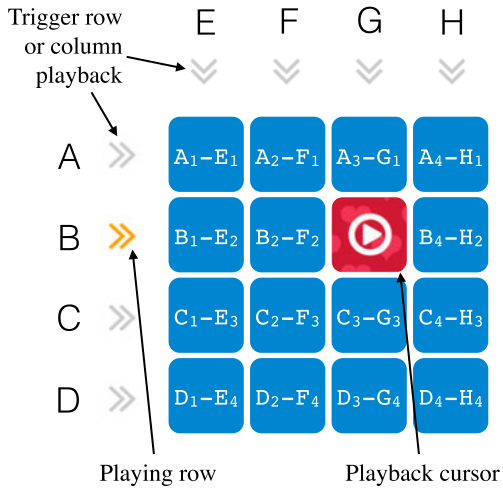


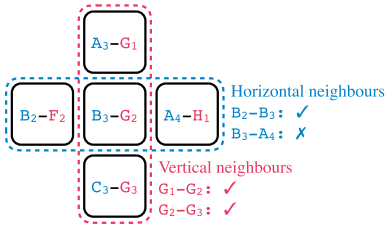
Fig. 2. Illustration of CrossSong Puzzle interface. In addition to indicated controls, clicking on two different tiles swaps their position. Row, column and tile labels are added for clarity, but do not appear in the interface. X_i indicates the ith part of excerpt X.

During gameplay, audio playback is continuous: the tiles play one after the other, first by row (from left to right, top to bottom), then by column. The tile currently being played is highlighted. Because all the tiles have the same duration and tempo, even in the initial random configuration there is a rhythmic coherence to the music.

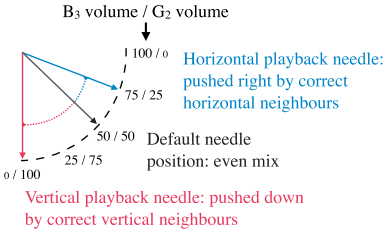
The player can perform two actions: first, they may click on any two tiles to swap their position. Second, they may click on arrows outside the grid to choose which row or column to play next after the current tile has finished playing. It is also possible to pause and resume the game. A link to a gameplay video is given in Figure 1. For beginners, solving a single puzzle takes roughly 10 min; an expert par time is closer to 3 min.

Normally, the two clips in each tile are played with equal loudness. However, as a hint and a reward for partial progress, the balance between the clips changes if the tile is positioned correctly with respect to its neighbours. The more correct neighbours, the more the mixing is reduced. The concept of 'relative cell correctness' is illustrated in Figure 3(a). In this example, the tile B_3-G_2 has one correct horizontal neighbour, since the tile B_2-F_2 belongs to its left in the solved puzzle. The impact of this arrangement is seen in Figure 3(b). When B_3-G_2 is played as part of the current row (horizontal playback), instead of the mix being 50/50, it will be 75% B_3 and 25% G_2. When played as part of the current column (vertical playback), since both vertical neighbours are correct, the mix will be 100% G_2 and 0% B_3. It does not matter if B_3-G_2 is in the correct position within the 4×4 grid; this audio mixing is based only on relative correctness.

We implemented the game as a web-based application. This has the advantage of making it instantly cross-platform: we have played it successfully on a desktop with a mouse, on a smartphone with a touch screen, and even on a large-format touch screen with multiple users (as pictured in Figure 9).



(a) Illustration of relative cell correctness.



(b) Illustration of how clips are mixed depending on correctness.

Fig. 3. Diagrams for how neighbour correctness is calculated for a given tile, B_3-G_2, and the resulting balance when played as part of a row or column.

Once a puzzle has been generated (discussed in Section 4), it is presented to the player in a JavaScript interface. We used the Web Audio API, allowing us to leverage the increasing capabilities of modern web browsers for interactive audio applications (Wyse & Subramanian, 2013).

4. Puzzle creation algorithm

As described in Section 2, AutoMashUpper estimates the mashability of two excerpts as a function of their harmonic, rhythmic and spectral compatibility, considering a range of possible transpositions. AutoMashUpper finds, for a given section of a song, the single best matching segment among a list of other songs. Our goal is different: to find a set of eight song excerpts, each of which can be divided into four equal-sized clips, such that, when arranged into a 4×4 grid, each combination of clips forms a good mash-up.

The problem is similar to generating a crossword puzzle grid: for that task, letters must be found which create acceptable words in each direction. However, a strict similarity function applies for letters—they are either the same or not—but no binary measure of acceptableness is available to us for music. The crossword generation problem, though seemingly straightforward relative to our task, has been researched for decades (e.g. see Ginsberg, Frank, Halpin, & Torrance, 1990). It is a complex search problem that is NP-complete (Engel, Holzer, Ruepp, & Sehnke, 2012).

Our primary obstacle is the vast space of combinations to search. Each excerpt can begin on any downbeat, meaning there are roughly 100 choices of excerpt in a typical song (this is the case for a 120BPM song that lasts 3:20). For 8 songs, this gives $100^8 = 10^{16}$ possible sets of excerpts. For each set, there are $8!/2 = 20160$ ways of arranging them in the 4×4 grid. (The factor of 2 reduction recognises that any arrangement and its matrix transpose are equivalent.) Finally, each excerpt may be transposed (in pitch) up to 3 semitones upwards or downwards, increasing the space by a factor of 10^7. (Precisely, it is a factor of $7^8 - 6^8$, or 4,085,185.) In total, this means 10^{28} different solutions—about 10,000 moles—clearly too many to search exhaustively.

Before explaining how we reduced this search space, here is the overall procedure for computing mashability, searching for an optimal mash-up, and processing the audio.

(1) Extract harmonic, rhythmic and spectral features from the 8 recordings (Davies et al., 2014)
(2) Estimate the mashability of all pairs of points in two different songs (Davies et al., 2014)
(3) Search for the choice of excerpts and their arrangement in a grid that maximises mashability
(4) Process audio clips to generate the puzzle grid
    (a) Apply time-stretching and pitch shift to match all excerpts using Rubber Band library (Cannam, 2012)
    (b) Match loudness of all excerpts using FFmpeg[11]

Feature processing (step 1 in the list above) requires roughly 14 s to analyse each song (based on an average 3-minute song). Step 2, computing the mashability, takes roughly 0.5 s per pair of songs, or 14 s overall for an 8-song puzzle. These steps can also be executed in advance if the library of music was previously known to the system. The audio processing using Rubber Band and FFmpeg takes about 10 s. The bottleneck is the incredible number of random sets of excerpts.

There are several dimensions to the search in step 3. First, we choose a set of 8 songs, labelled P through W; we are assuming that this is fixed by the user. Second, we select an excerpt from each song; let P^p represent the choice of segment p from song P. Third, we fit the songs into the grid; i.e. we choose a mapping of $\{P, Q, \ldots, W\} \rightarrow \{A, B, \ldots, H\}$, where $\{A, B, C, D\}$ are the rows and $\{E, F, G, H\}$ are the columns, as before. For a choice of segment indices $\{p, q, \ldots, w\}$, it is computationally cheap to find the optimal mapping in the grid. Thus, in the search algorithm, we focus on finding the best choice of song segments to use.

In step 3, we estimate the quality of a solution grid by taking the sum of each tile's mashability $M(X, Y, d_{XY})$. Here, $X \in \{A, B, C, D\}$ is a row excerpts, and $Y \in \{E, F, G, H\}$ the column excerpts, and d_{XY} represents the delay required for the given combination of row and column. For example, for tile (A, G), we are interested in the mashability when excerpt G is delayed by half its length with respect to excerpt A; this is because, as we saw in Figure 2, G_1 (the first bar of G) coincides with A_3.

[11] http://www.ffmpeg.org.

4.1 Search optimisations

There are two ways to make the search space less daunting: to reduce the search space, and to improve how the search algorithm scans the space. Our search optimisations include both techniques.

4.1.1 Eliminating segments

We have already limited our search space to passages that begin on downbeats. We can further reduce the search space by restricting ourselves to excerpts that begin at one of the section boundaries estimated by AutoMashUpper, which uses a variation of the classic Foote algorithm (Foote, 2000). This restriction increases the odds that each excerpt will be an intact phrase of a song, and reduces the 100 or so excerpts in each song to roughly 20 to 30.

We can reduce the space even more by ignoring repeated segments. If a structural analysis predicts that two segments are the same (for example, that both are choruses), then it is not necessary to consider both in our search. What savings are possible with this technique? In the SALAMI data-set, the median number of large-scale segments with unique labels is 4 per annotation (Smith, Burgoyne, Fujinaga, De Roure, & Downie, 2011). (For small-scale segments, the median is 7.) Methods of detecting repetition are countless and range in complexity; for a recent review, see (Müller, 2015). For speed and simplicity, we can use k-means to cluster the potential excerpts based on the features we have already computed: the harmonic, rhythmic and spectral features used by the mashability algorithm. For each estimated cluster, we retain the excerpt that is closest to the cluster centroid, and ignore the others. After this step, the number of excerpts is reduced from 15 or 30 to k, which we can set freely. If the 100 potential excerpts are reduced to 8, the search space shrinks by more than 10^8.

4.1.2 Eliminating transpositions

Our next optimisation is to only consider, for each pair of excerpts, the transposition that gives the optimal mashability—even though this may not be the transposition used in the final puzzle. This reduces the search space by a factor of 10^7, but it can clearly lead to problems: the final grid will require that all the clips be transposed to match each other, but these optimal transpositions might be infeasible. For example, suppose we have chosen clips for slots A, B, E and F on the basis of their optimal mashability, disregarding the required transpositions. We then transpose E_1 to match to A_1, F_1 to match A_2 and B_1 to match E_2. However, this fixes the transpositions of B_2 and F_2, and the result may be dissonant.

In order to mitigate this, we compute mashability not between individual fragments (such as A_2 and F_1), but between full excerpts (i.e. between A and F, with F delayed by one quarter—i.e. $M(A, F, d_{AF})$). This creates some mutual dependence in the mashability values. In the previous example, we know that B_2 and F_2 will match as long as B_1 and F_1 match. Assuming all the mashability values are high, we know that B_1 matches E_2, which matches A_2, which matches F_1. Hence, to the extent that harmonic compatibility is transitive, we can use a greedy approach without worrying too much about conflicts in transpositions.

Note that the problem above applies only to harmonic mashability. Rhythmic and spectral mashability are not affected by this simplification, since neither are expected to be affected by small transpositions in pitch.

4.1.3 Stochastic search

Since an exhaustive search would take too long, our aim is to find the highest quality solution in a fixed amount of time. Stochastic search strategies, such as simulated annealing, are designed to accomplish this. Simulated annealing strikes a balance between random search—always checking random new solutions, exploring the search space broadly— and greedy search, in which we focus on a narrow part of the search space that seems promising. Crucial, then, is that we have a concept of neighbouring solutions: if a solution is a vector $\{p, q, \ldots, w\}$ of segment indices, then we can generate a neighbour by changing a single one of these values.

At each step in our search, we can either choose a random solution or choose the neighbour of a good existing solution. Briefly put, the trick of simulated annealing is to make this choice probabilistically, and to transition smoothly from a tendency for random steps at the beginning, to a tendency for greedy steps near the end of the allotted time. The effect is illustrated in Figure 4. The figure depicts a typical set of 100 solutions, ranked in order of mashability (this is the black line). In a random search, any unexamined solution could be tested next; in a greedy search, we would use the current optimal solution (the one at the far left) as our basis and choose some unexamined neighbour of it to look at next. If no neighbours remain, we 'pop out' of the local maximum and look along the next-best path. But we can also use the mashability scores as the relative probability of choosing each solution as a basis. Simulated annealing does this by letting the relative probability of choosing a basis i be $p_i = \exp(\frac{M_i - M_{\max}}{t})$, with M_i the mashability of the basis, M_{max} the greatest mashability found so far, and t be the 'temperature' which cools from 1 to 0. As $t \to 0$, the $p_i \to 0$ if $M_i < M_{\max}$. The plot shows us the probability curves for evenly spaced values of t starting at 1 (red) and ending at 0 (blue).

4.1.4 Local sampling

Earlier we noted that for every choice of excerpts, there are $8!/2 = 20160$ possible arrangements to consider. However, it turns out that these arrangements might vary little in terms of mashability. Figure 5 plots the maximum mashability found in an exhaustive search of grid arrangements compared to the average mashability found in a random set of 50 arrangements. The close correlation suggests that sampling a few solutions for each choice of excerpts may be more efficient than exhaustive local search.

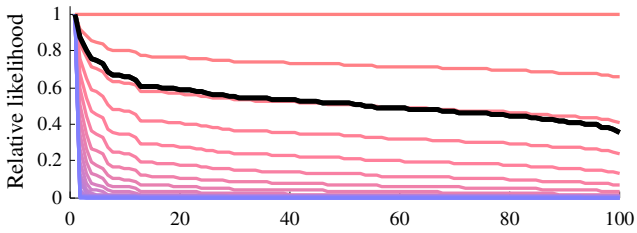


Fig. 4. In black: relative mashability (y-axis) of typical set of 100 solutions, sorted by rank (x-axis). In colour: relative likelihood (y-axis) of continuing search from a given ranked solution, for various time steps from beginning (red) to end (blue) of allotted time.

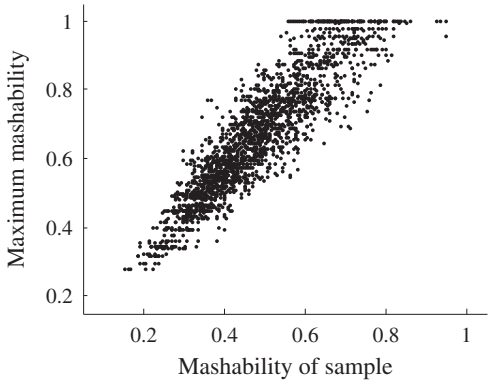


Fig. 5. Maximum mashability of all 20160 grid arrangements (y-axis) vs. best mashability in a random sub-sample of 50 arrangements (x-axis). Data obtained from 100 random choices of song excerpts for each of 20 random sets of songs (mashability normalised to maximum discovered in each set). The strong correlation indicates that exhaustive local search may not be necessary.

4.2 Runtime experiment

High-dimensional search spaces can be complex, and there is no certainty that any search heuristic will be successful. We tested the effectiveness of our proposed search optimisations in a runtime experiment. We used a full-factorial design with the following settings: (1) checklist size: i.e. how many arrangements to test for each choice of excerpts (50, or the full 20160); (2) search style: random, greedy or simulated annealing; (3) whether to use the structural analysis (with $k = 8$) or none at all; and (4) runtime, which we set as either a rush job (10 s) or potential background job (3 min).

For each setting, we ran trials on eight combinations of songs from the RWC Music Database (Music Genre, Goto, Hashiguchi, Nishimura, & Oka, 2003). Because mashability (which has arbitrary units) strongly depended on the combination of songs—for one set, the maximum value was 29, for another, 50—we have re-scaled the values by the maximum and minimum discovered values for each set. The results of the experiment are shown in Figure 6.

Naturally, the longer the runtime, the better the results. All of the other factors were also found to have a significant effect on the mashability of the best grid. Moreover, we found a strong interaction between all factors and the available runtime.

First, we found that structural analysis was not helpful to the search process. The analysis shrinks the search space, but, disappointingly, seems to do so at the cost of good solutions. The difference was minor for short searches, but quite large for long searches.

On the other hand, we found that the other two heuristics both led to gains—but only during short searches. In a short search, greedy was better than random, and simulated annealing better still. However, in a long search, greedy outperformed annealing (and both were much better than random). Perhaps this is because the greedy approach, over a long time, is actually able to exhaustively search local maxima and begin searching for others.

Lastly, the local search strategy was very important in short searches. By sampling instead of using exhaustive local search, the algorithm could test more excerpt combinations. However, in a long search where the algorithm had more time to find the good combinations, the exhaustive local search could find the best solutions of all. Also, a simple refinement of the algorithm—to use the small checklist size, and at the end of the allotted time perform an exhaustive grid search on the best grid so far—would give the best of both strategies.

While the annealing and local sampling search heuristics will be useful strategies for time-limited applications, like when making a puzzle out of a user's music, more straightforward, exhaustive approaches would be best for generating puzzles offline.

In future search systems, we may wish to consider different combinations of songs, since, as stated earlier, this had a big impact on mashability. Considering the choice of songs would exponentially increase the size of the search space. However, based on our experiment, we would expect the local sampling and greedy heuristic to speed this considerably.

5. Design development

A puzzle creator has two contradictory goals: first, to confront the solver with a problem that is difficult to solve; and second, to ensure that the solver is eventually successful (Gottlieb, 1998). Hence, our first goal was to iteratively develop our puzzle design until we felt it struck the correct balance between posing no challenge and posing an insurmountable one. In the process, we also kept in mind two design criteria that are supported by the popular concept of 'flow' (Csikszentmihalyi, 1990), which seeks to explain why certain activities are more engaging than others. Namely, we felt that the player's goals should be clear and manageable, and that feedback should be frequent and useful.

In this section, we describe the sequence of puzzle designs we developed, including the pros and cons of each. The assessments at this early stage of development are based on the authors' own experience in testing versions of the game, as well as those of colleagues we recruited to test it. Since having fresh eyes is essential to effective playtesting, we had new colleagues test each iteration. A more formal evaluation of the game is discussed in Section 6.

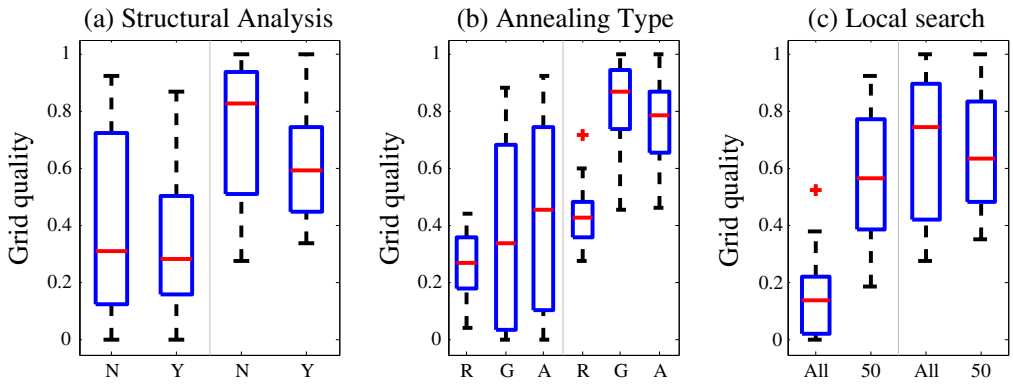


Fig. 6. Boxplot of mashability of grids (scaled to global maximum found) in a 10-second search (left half of each plot) or 3-minute search (right half). The three plots compare: (a) structural analysis not applied (N) vs. applied with $k = 8$ (Y); (b) random, greedy or annealing search; (c) local search strategy: exhaustive local search of all 20160 solutions vs. a brief search of 50 solutions.

Our iterations primarily affected three aspects of the puzzle: first, the balance of visual and auditory hints given; second, the way that the puzzle confirmed the progress of the solver; and third, how the listener's familiarity with the musical excerpts was handled.

5.1 Version 1: Initial prototype

Our initial prototype worked as described in Section 3. All of the basic gameplay elements of this version—the swapping of tiles, the control of row and column playback and the fading audio hint based on row correctness illustrated in Figure 3—were retained in future versions. However, there was a slight difference: only the top-left-most tile was fixed.

The puzzle was enjoyable to work on, but it was only solvable by those who knew the music beforehand. None of those who tested this version without knowing any of the music solved it; one user even spent 10 min without being certain of the relative position of any tiles, and was very discouraged.

We concluded that more fixed tiles would be needed to give players a decent starting point. Moreover, at least one of these tiles would need to be off the main diagonal: we had not yet realised that the transpose arrangement of tiles was logically sound but not recognised by the system as correct, nor reinforced by the audio hints. This meant that if users re-assembled the first 'across' song, but arranged the tiles in the column instead, they would get no confirmation—auditory or otherwise—that they had done the right thing.

5.2 Version 2: Adding hints

Our second version added two more fixed tiles to get started in the upper-left corner. This meant that solvers had an 'in' to start the puzzle: i.e. there was a place where solvers could easily make progress, even if they were unfamiliar with the music or how the puzzle worked. Feeling that this was not sufficient to make the puzzle feasible, we also added strong visual hints to support the audio: the relative correctness of every tile was shown by displaying icons at the boundary with the correct neighbour (these are the hearts seen in Figure 7(a)).

Unfortunately, the visual hints made progress too rapid: once a few tiles had been placed in the correct order, the rest of the puzzle could easily be solved visually by checking different configurations through trial and error. Although we agreed that some visual confirmation of one's progress was needed, this version took the focus of the logic away from the audio, defeating the intent of the puzzle. The ideal visual hint would reinforce the auditory hint without immediately adding any new information.

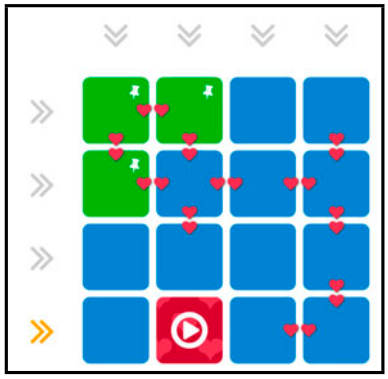
5.3 Version 3: Refining visual hints

Our solution was to animate the background of the currently playing tile: we added a textured background that flows in the direction of the arrow in Figure 3(b). For example, if no neighbours are correct, the background flows in a south-easterly direction; if both horizontal neighbours are correct during horizontal playback, the background flows eastward. Thus, the solver gets a visual confirmation of the relative correctness of the tile, but without extra clues about which neighbouring tiles are correct. Also, the visual clue is only available *while* the solver listens to the tile, so solving it visually is too slow to be effective. Although it is still technically possible to solve it using only visual hints, the necessary brute-force method is slow and tedious.

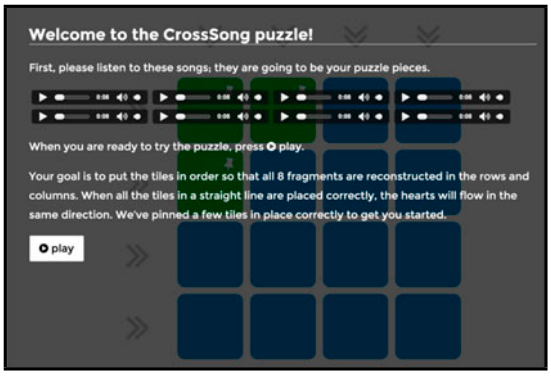
Those testing this version reported that the puzzle was still too difficult, for two reasons. First, mentally keeping track of the tiles was taxing, and it was easy to undo one's progress: for example, one might sort several similar tiles into a single row, but then forget which row it is, or accidentally swap a tile away and lose track of it. Second, the puzzle was still very difficult for first-time listeners; many of the mash-ups blended well enough that it was hard to tell which parts of a tile belonged to which song!

5.4 Version 4: Improving usability

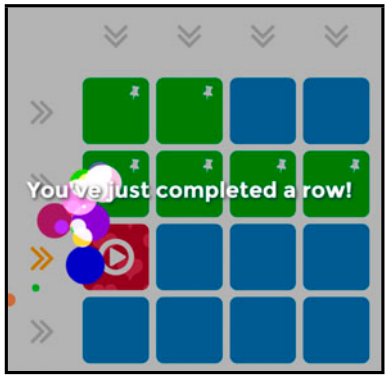
We next added two features to make the game more user-friendly. First, following the example of Hansen et al. (2013), we added a welcome screen (see 7(b)) where solvers were



(a) Visual hints added to Version 2



(b) Welcome screen, added to Version 4



(c) Row confirmation screen, added to Version 4

Fig. 7. Screenshots of development versions of CrossSong.

allowed to listen to each of the eight excerpts separately before solving the puzzle—just like jigsaw puzzle solvers can look at the picture on the box first.

Second, we added a row-confirmation feature (Figure 7(c)). If all the tiles in a single row or column are placed in their correct position, a congratulatory message appears, and the tiles become fixed in place—but only *after* that full row (or column) is played. This way, randomly shuffling tiles is still a fruitless approach. Fixing the tiles in place prevents undoing one's work but also serves as an encouraging confirmation of partial progress, which is a feature of many engaging puzzles. A typical sequence of gameplay steps leading up to this row confirmation event is depicted in Figure 8.

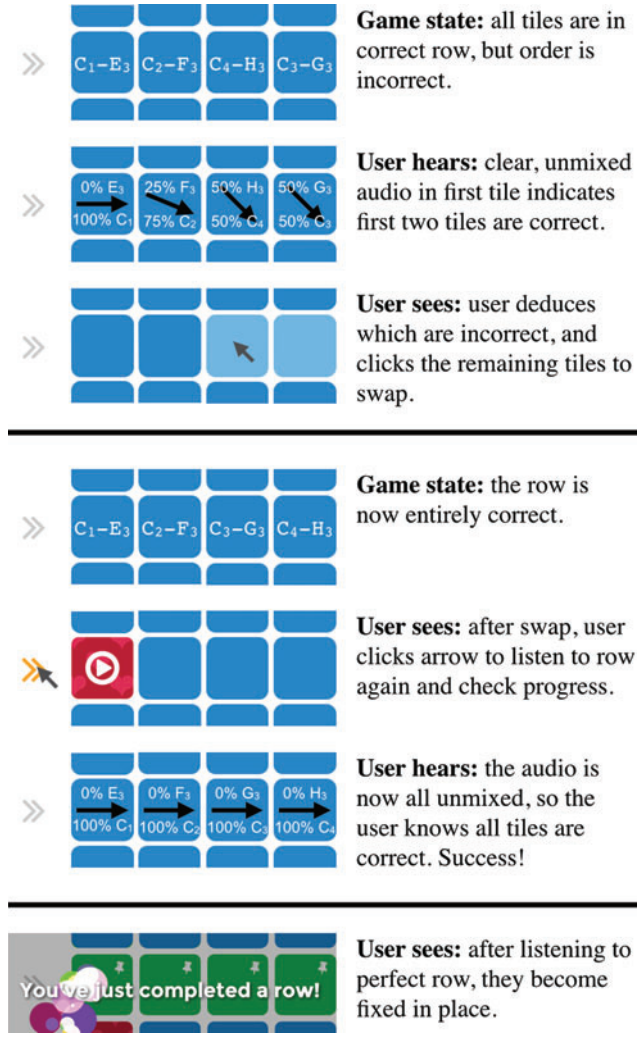


Fig. 8. Depiction of a typical gameplay sequence. In the top part, the audio cues help the user identify which tiles arranged incorrectly. In the middle part, the user listens to the new arrangement. The bottom part shows the visual feedback provided to the user.

This version of the puzzle, we felt, was fun to play and had most of the qualities we sought: it combined a need for careful listening with logical deduction, and although supported by visual hints, the visual hints did not dominate the puzzle-solving experience. The puzzle also sets up a series of goals (the row and column confirmations) that are achievable whether one is playing with one's favourite songs, or someone else's. The game is available to play online.[12]

5.5 Version 5: Controlling difficulty

Although most testing issues were evident after a few test-runs, some only became apparent over multiple trials. Overall, we had the impression that the game was too hard, and possibly too fast-paced.

To reduce difficulty, we tried increasing the unit length of the tiles. Previously, we had generated puzzles where each

[12] https://staff.aist.go.jp/jun.kato/CrossSong/.

tile was four beats long. Doubling this to 8, we found that the mash-ups in each tile could be understood more clearly. Over eight beats, with more independent activity from the two songs, it was easier to parse the components of the mash-up and recognise the music. Moreover, we found the less chaotic mash-ups more pleasing to listen to.

We also changed the audio mixing levels of the two mash-ups. In previous versions, the mixing level (which depends on a tile's relative correctness—recall Figure 3) progressed from 50/50 to 100/0, rendering one song inaudible. We changed 100/0 to 70/30 (and shifted the midpoint accordingly), so that one song became very quiet but was still perceptible. Although we previously enjoyed the pay-off of hearing perfectly-isolated excerpts in a completed puzzle, the isolation could make progress more difficult during solving. Being able to faintly hear the 'cross-songs'—e.g. hearing parts of the 'down' songs when playing a row that is entirely correct—was a useful solving aid.

Finally, we created a number of different tile layouts. The variety is enjoyable when playing several levels in succession, and fixing more tiles in place is a simple way to reduce the difficulty of the game.

5.6 Future versions

Although the puzzle has been conceived as a single-player game, as an interface it can accommodate alternative modes of interaction. It is straightforward to use as a multiplayer game: Figure 9 depicts two players playing cooperatively on a large touch screen device. In this setting, each solver can choose four songs to include in the diagram. The search algorithm can add an extra constraint to ensure that one player's choices appear in the rows, and the other player's songs in the columns; this actually simplifies the search by a few orders of magnitude.

The interface is also a rudimentary but engaging remix device. It is fun to ignore the game goals and use the interface instead to arrange sounds and control their playback. In this way, the interface resembles automatic remix engines like *Adventure Machine*[13] and *Girl Talk in a Box*.[14]

6. Evaluation

Having arrived at a game design that satisfied us in playtesting, we evaluated it in a controlled trial with 10 participants. One goal of the evaluation is to assess the usability of the system, and to uncover any remaining issues with the design. For usability studies, 5 participants is often regarded as sufficient (Greenberg & Buxton, 2008). However, another goal of the evaluation is to discern whether playing the game has any impact on players, such as stoking an interest in the music used by the puzzle, or in mash-ups generally. To answer these questions definitively, a larger sample size would be needed to perform statistical tests, but with 10 participants we can at least get an indication.

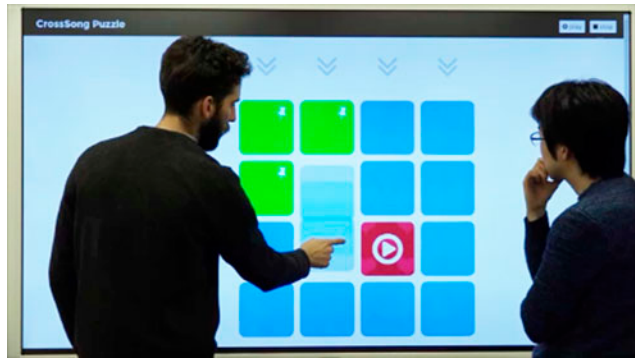


Fig. 9. CrossSong Puzzle with two users working cooperatively.

6.1 Procedure

Participants filled out a background questionnaire, completed a sequence of puzzles, and then an evaluation questionnaire. The study took about 1 h to complete. Participants completed the study online (from the privacy of their home or workplace), and were asked to use headphones. The full experiment sequence was as follows:

(1) Introductory sequence (10 min)
   (a) Introduction and overview of study
   (b) Preliminary questionnaire (musical sophistication, genre preference, game preference)
   (c) Tutorial video

(2) CrossSong puzzles (35 min)
   (a) Tutorial sequence
      (i) 2×2 puzzle
      (ii) 3×3 puzzle
      (iii) 4×4 puzzle with one row or column left to solve
   (b) CrossSong puzzles
      (i) Sequence A:
      Puzzle X: 4×4 using music from the RWC Music Database (Popular Music, Goto, Hashiguchi, Nishimura, & Oka, 2002).
      Puzzle Y: 4×4 using music from the RWC Music Database (Jazz Music, Goto et al., 2002) and Western pop music.
      (ii) Sequence B:
      Puzzle Z: 4×4 using Western pop, indie and electronic music
      Puzzle X: same as above.

(3) Evaluation survey (15 min)

[13] http://www.madeon.fr/adventuremachine.
[14] http://static.echonest.com/girltalkinabox/.

For each participant, the background survey assessed: their musical sophistication, using the standard Goldsmiths MSI short test (Müllensiefen, Gingras, Musil, & Stewart, 2014); their preference for different musical genres, with a set of questions modelled on the Short Test of Musical Preference (Rentfrow & Gosling, 2003), but adapted for a Japanese audience; and their preference for various types of games, including rhythm video games, puzzle video games, logic puzzles and crossword puzzles. We used version 1.0 of the self-report MSI questionnaire, abridged to only those 18 questions that load onto the 'General Sophistication' factor.

From our experience introducing the puzzle to others (reported in the previous section), we knew that the game was not self-explanatory: players needed at least some explanation of the game mechanics. During the development of the game, this had been explained to playtesters in person, but for our evaluation, we needed a succinct script that could be consistent between participants and available online. We thus created a two-minute video that explained the goal and the main mechanics of the puzzle, i.e. the controls, the way that correct rows lock in place, and the hints at partial progress provided by the background. Believing that simpler versions of the puzzle would allow new players to absorb the game mechanics without feeling overwhelmed, we also crafted a short tutorial sequence (see Figure 10). It begins with simple 2×2 and 3×3 versions of the puzzle; the 2×2 is usually solved with one move, but introduces players to the sound of the game and the controls. For the 3×3, we explain the most basic sound-based solving strategy, which is to pick a nearly completed row, 'focus on its sound, and then search for the missing tile in the rest of the grid.' Then, in a 4×4 puzzle which has all the tiles in the correct place except for one row, we explain the usefulness of the visual cues: by playing the columns, the correctness of any tile can be immediately deduced from the background.

The final step was a follow-up questionnaire of 20 questions that addressed the goals of the evaluation. The questions assessed: if they liked the game (2 questions); if the difficulty was appropriate (2 questions); if the tutorial exercises were useful (2 questions); if the music was enjoyable (3 questions); which was more important among the audio and visual hints, or whether pure luck dominated (3 questions); if the interface was user-friendly (4 questions); and if the game had an impact on the user (4 questions). Several questions were borrowed from the widely-used System Usability Scale (Brooke, 1996), and the other questions were written in a similar style. The questions are listed in Table 1.

The study was conducted in Japan, so the questionnaires, instructions, tutorial video, and so forth were all presented in Japanese. Only the basic messages within the game (e.g. the 'congratulations' message, the 'play' button and the welcome screen) were left in English. However, the participants were skilled enough in English that this did not pose a problem. The Japanese translations of the Goldsmiths MSI questions will be made available for other researchers online. They were translated by the 2nd and 3rd authors, who are both fluent in Japanese and English.

6.2 Participants

Of the 10 participants, 2 were female and the median age was 30 years old, with a range from 19–34. On the 18-question Goldsmiths MSI questionnaire, the median score was 72, which coincidentally is the exact midpoint of the scale—although it corresponds to the 32nd percentile compared to the general population (Müllensiefen, Gingras, Stewart, & Musil, 2013). The scores fell in the range 43–100, or between the 4th and 79th percentile.

The participants also rated on a seven-point Likert scale how much they enjoyed various genres of music and types of puzzles and games. All but one of the 10 participants indicated that they at least 'somewhat liked' J-Pop music, Seven of the 12 genres we asked about were popular genres; when pooling all the popular genres, the average rating from every participant was greater than 4. Preference for puzzles and games was less unanimous: out of 10 participants, 7 gave average Likert ratings above 4, but within this group of people who enjoyed puzzles, average ratings fell between 5.75 and 6.75.

6.3 Results

The average user responses for the evaluation questionnaire are shown in Figure 11. The responses for related questions have been grouped together. The results indicate that the puzzles were enjoyed by most participants (*liking*), and that all participants found the interface to be intuitive and easy to use (*interface quality*). Also, most players agreed that listening closely was the most important skill, although the visual hints were also needed (see Q10 and Q11 in Table 1). Thus our main design goals seemed to be successfully achieved.

Responses were more mixed on the questions of *feasibility* and *tutorial usefulness*. Four of the participants agreed that the puzzles were too hard, and only four participants reported never feeling lost or discouraged. While players being lost is not necessarily a bug for a puzzle game—we want to challenge the players—we do not want to discourage them. On the other hand, six participants felt that the tutorial exercises were boring or not necessary. This was interesting to hear, because we were quite certain, based on our informal tests, that the tutorials *were* essential to the puzzle's feasibility: during the prototyping stage, before we introduced the video and tutorial, new solvers were usually unable to make any progress in the 4×4 puzzles—much less solve them in 10 min or less—without some assistance. In fact, this evaluation was the first test in which all participants were able to complete all the puzzles without supervision, and in the expected amount of time.

Participants also seemed to find the *music enjoyable*. Beyond this, we hoped that listening carefully to the short excerpts might inspire in the participants a desire to hear the rest of the songs (*future engagement*), but the results here are mixed. However, nearly all participants agreed that they would like to play the game again with music of their choosing (Q19 in Table 1),

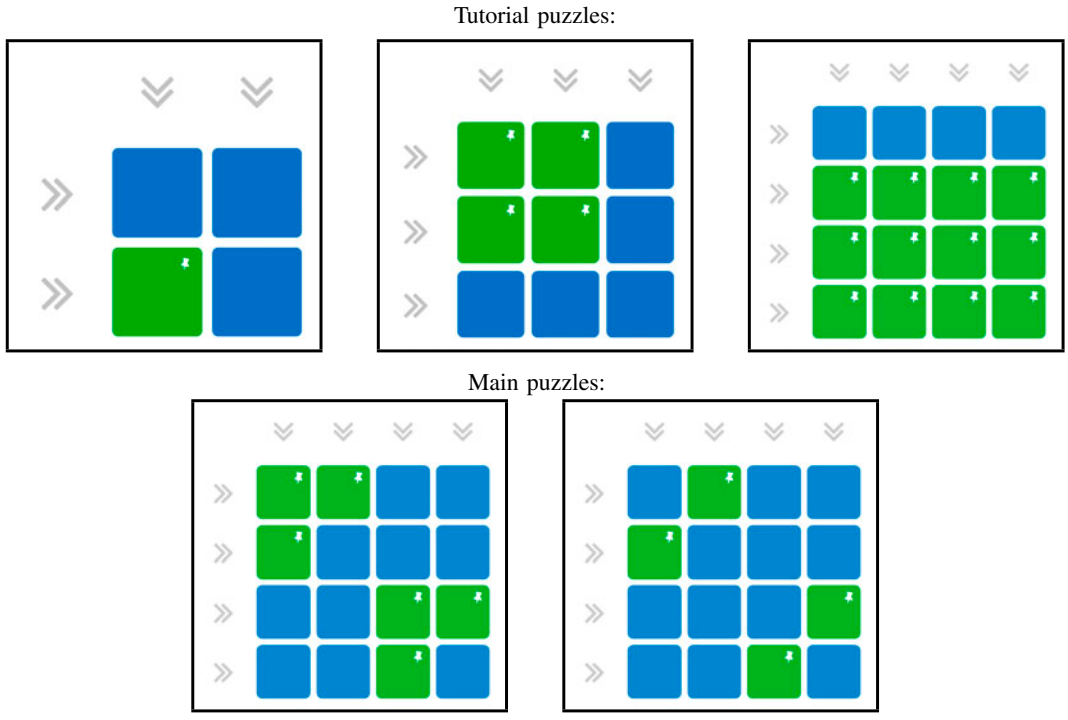


Fig. 10. Screenshots of opening layouts for sequence of tutorial puzzles.

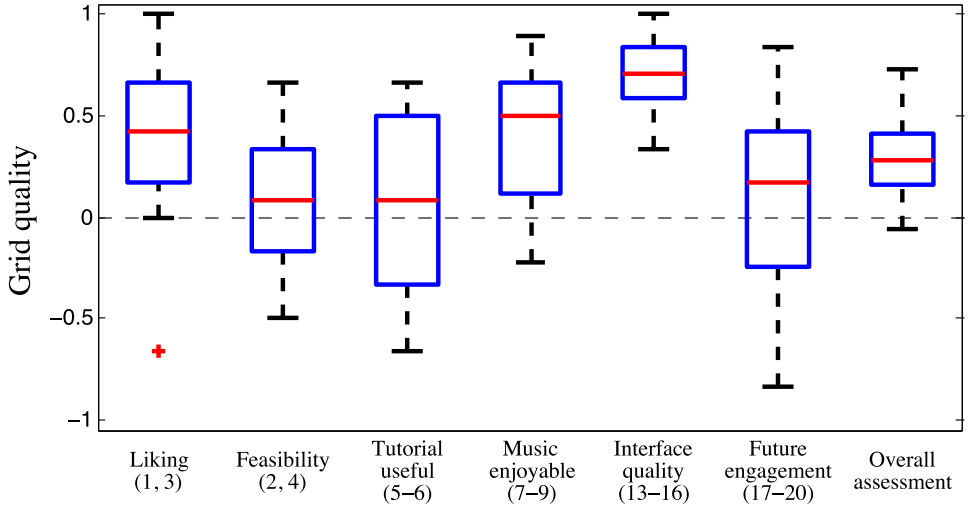


Fig. 11. Distribution of survey responses, grouped by factor. Likert scale answers (1–7) normalised to (−1, 1). Numbers in brackets indicate the questions included in the factor; see Table 1 for questions and detailed responses. (Responses to negatively phrased questions were inverted before being grouped in this summary figure.)

We anticipated that the experience of the participants might depend on their musical sophistication, or their stated enjoyment of pop music or games and puzzles. There were too few responses to apply statistical tests, but in any case we did not observe any hint of a relationship between the participants' background and any of the evaluation factors. The only exception was an apparently strong correlation between how much participants reported enjoying games and how much they enjoyed this one (Figure 12). This may have been due to the outlier on question #1—the person who did not enjoy CrossSong—also happening to be the person who reported the lowest enjoyment of games.

We also solicited written feedback from participants. They made several suggestions for improvements to the game: some longed for a more 'complete' interface to enhance the user experience, such as indicating a player's 'score', or otherwise tracking their progress. The number of tile swaps made and the time elapsed could be used to generate a player's score (with

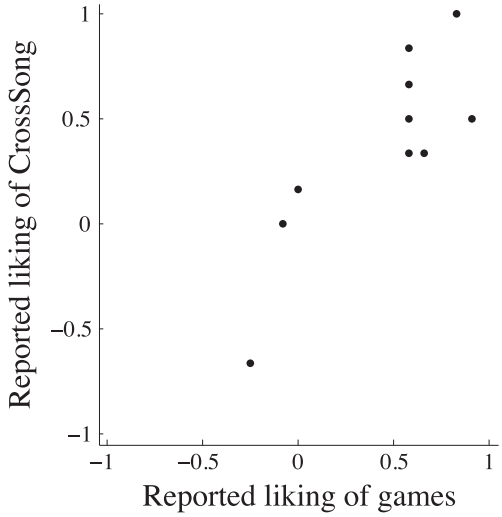


Fig. 12. Scatter plot of reported liking of games generally (on background survey) and reported liking of CrossSong.

smaller values of each giving higher scores). Others pointed out shortcomings of the interface that would be simple to fix, like the hard-to-find 'pause' button (users had to scroll down to reach it), or the way users are instructed to listen to the full excerpts first, which can be time-consuming. Although this was optional, the experiment setting implied that it was mandatory.

But overall, their comments indicated that users considered the game to be a very interesting idea. One user praised the pay-off of the un-mixing: he did not notice which songs were used when mashed up, and felt it was very interesting when, upon solving, the reconstructed song would get prominent focus.

Encouragingly, novice users without extensive knowledge of music wrote that they enjoyed the game and felt confident playing it. Recall that musical sophistication was not found to correlate with any of a player's survey answers. Some players reported listening to the music more carefully than they had before, and the phrases they used in their comments—focusing on the 'components of the songs', or noticing the 'typical sound of musical instruments'—suggests that CrossSong may be an interesting way to get musical novices to think about music more critically.

7. Discussion

In Section 1, we noted a challenge to designing an effective musical puzzle: the fact that music happens in time, whereas puzzles can be solved at an arbitrary pace. Reflecting on the evolution of the design after multiple rounds of playtesting and redesign, we realise now that the success of a music puzzle hinges on many more issues, including these three points:

(1) **Balancing visual and auditory hints.**
The goal of a music-based puzzle should be to challenge a solver's ability to listen carefully and to think about music. On the other hand, a puzzle needs some kind of visual component for the user to interact with, so a display is necessary to let the solver confirm or track their progress. However, if all information relevant to the puzzle is displayed visually, this defeats the intent to make a listening-based puzzle. Thus the main challenge in designing a puzzle to engage musical thinking may be to figure out how to introduce visual hints while confining them to a supporting role. The key to this balance in CrossSong was to make the visual hint (i.e. the background animation direction) available only while the music was being played. This kept the gameplay focus on listening to the music, but allowed solvers to make some progress even when they were otherwise lost.

(2) **Using multiple mechanics to modulate difficulty.**
It must be possible to vary the difficulty of the puzzles: simpler puzzles are needed to show a new player the ropes, and more difficult puzzles are needed to maintain their interest. Many factors affect puzzle difficulty, but not all of them can be controlled. Some factors are baked into the design, and changing them would affect difficulty too drastically: these include the balance of visual and auditory hints, and the way that correct rows lock into place. Some factors depend on the user, including their familiarity with the music and their listening skills. Thus, it is important that the puzzle design accommodate at least a few parameters that can be easily tweaked to control difficulty: in the case of CrossSong, changing the size of the grid and the arrangement of fixed tiles are two ways to control difficulty predictably.

(3) **Designing levels automatically.**
If users are able to choose what music to use in a puzzle, this gives them another way to control the difficulty of puzzles. The potential to play CrossSong with music of one's choosing was part of the game's appeal, and participants in the evaluation agreed. But user-selected music demands procedurally generated puzzles. That is: automatic level design is a requirement. In the case of CrossSong, the ability to generate levels depended on an efficient search algorithm that we applied to existing mashability estimation techniques. Without this infrastructure, we would not have been able to rapidly develop the prototype, nor produce puzzles tailored to the users.

Eran Egozy, co-founder of Harmonix (the company that created the music games *Guitar Hero, Rock Band, Fantasia: Music Evolved,* and others) has written that throughout his career, he has often held the same goal in mind: to engage players while allowing them to be musically expressive (Egozy, 2016). However, it took almost 20 years of developing games for him to judge that he had come close to realising that. Above, we offered the design lessons we learned in the course of developing and testing CrossSong, which we believe are applicable to any music-based puzzle. However, ours remains

Table 1. Raw average and standard deviation of survey responses for each question. The neutral midpoint of the seven-point Likert scale was 4; percent favourable responses thus gives the fraction of responses strictly greater than 4, or, for questions phrased negatively and marked with a '–', strictly less than 4.

Question	Average response	Standard deviation	% Favorable responses
(1) Playing CrossSong was fun.	5.6	1.36	90%
(2) (–) The puzzles were too hard.	4.0	1.61	40%
(3) I would like to play the game again.	4.6	1.43	50%
(4) (–) At some point while solving, I felt lost or discouraged.	3.7	1.42	40%
(5) (–) Solving the exercises was boring.	4.2	1.66	40%
(6) (–) I would have been able to solve the puzzles without doing the exercises first.	3.7	1.90	50%
(7) The combination of sounds in the mash-ups was often interesting.	4.9	1.45	50%
(8) (–) I did not enjoy listening to the music while playing.	2.5	1.63	80%
(9) I liked the music that was used in the puzzles.	5.2	1.60	70%
(10) I mainly solved the puzzles by listening closely to the music.	5.5	1.75	70%
(11) (–) I mainly solved the puzzles by focusing on the background animation hints.	4.3	1.73	30%
(12) Completing a CrossSong puzzle depends mainly on skill, not on luck.	4.4	1.43	50%
(13) The interface was easy to use.	6.3	0.78	100%
(14) (–) I found the interface to be very complex.	1.9	0.83	90%
(15) I felt confident using the interface.	5.7	1.10	80%
(16) (–) I needed to learn a lot of things before I could get going with CrossSong.	1.8	0.75	100%
(17) About the songs in the game I had never heard before: after playing the game, I would like to hear the rest of these songs.	4.1	2.07	50%
(18) About the songs I had heard before: after playing the game, I would like to hear these songs again.	3.7	1.90	30%
(19) I would like to play the game with music of my choosing.	5.0	1.95	80%
(20) I would like to hear more mash-ups.	4.3	1.73	30%

a first effort at creating a real-time music puzzle, and we anticipate new insights will come with developing more designs.

8. Conclusion and future work

We have proposed a novel type of puzzle, the CrossSong, which aims to combine the pattern-learning and pattern-seeking joys of music and puzzles. We have developed an algorithm for generating puzzles from music provided by a user, and an interface for solving them. The software allows (and solving the puzzle requires) the user to explore and to analyze a set of original mash-ups in real time as they are played. As we iteratively refined the design, we realised that the main challenge was to balance game mechanics that encourage the player to listen carefully to the music, with visual aids that can confirm the player's progress and make the task more feasible.

Participants in a small evaluation, regardless of their musical expertise, seemed to enjoy the game and expressed interest in playing it again with music of their choosing. To prolong interest in the game even further, we would like to demand different modes of musical thinking from solvers by varying the solving mechanics. As we indicated in the tutorial, certain listening strategies are more efficient in certain layouts, so to some extent, this variety can be achieved by manipulating parameters like the size and shape of the grid, the arrangement of fixed tiles and the duration of the tiles. But since the interface is so flexible, many more variations could be imagined. For example, the way the mixing works (cf. Figure 3) could be inverted: each tile could become *more* instead of less mixed as its neighbours became more correct. This version might become harder as the game progresses instead of easier. Another possibility is to treat the excerpts with harmonic-percussive source separation, and manipulate the mixing of these components to focus solvers' attention on rhythmic or harmonic aspects of the music. Creating, testing and evaluating new game mechanics is the future work we plan to pursue.

Acknowledgements

We would like to thank Matthew Davies for his original implementation of AutoMashUpper Davies et al. (2014).

Funding

This work was supported in part by OngaCREST, CREST, JST; Core Research for Evolutional Science and Technology.

ORCID

Jordan B.L. Smith http://orcid.org/0000-0002-0316-1235
Jun Kato http://orcid.org/0000-0003-4832-8024
Satoru Fukayama http://orcid.org/0000-0001-6506-2796
Masataka Goto http://orcid.org/0000-0003-1167-0977

References

Brooke, J. (1996). SUS: A 'quick and dirty' usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & I. L. McClelland (Eds.), *Usability Evaluation in Industry* (pp. 189–194). London, UK: Taylor and Francis.

Cannam, C. (2012). Rubber band library. Software released under GNU General Public License (version 1.8.1).

Ciconia, J. (1993). *De Proportionibus*. In O. B. Ellsworth (Ed.). Lincoln, NE: University of Nebraska Press.

Crute, T. D. (2010). Effective use of games and puzzles in the chemistry classroom. In *Making chemistry relevant: Strategies for including all students in a learner-sensitive classroom environment* (pp. 267–281). Hoboken, NJ: John Wiley & Sons Inc.

Csikszentmihalyi, M. (1990). *Flow: The psychology of optimal experience*. New York, NY, USA: Harper and Row.

Davies, M. E. P., Hamel, P., Yoshii, K., & Goto, M. (2014). AutoMashUpper: Automatic creation of multi-song music mashups. *IEEE Transactions on Audio, Speech, and Language Processing, 22*, 1726–1737.

Dubus, G., Hansen, K. F., & Bresin, R. (2012). *An overview of sound and music applications for Android available on the market*. Proceedings of the Sound and Music Computing Conference (pp. 541–546), Copenhagen, Denmark.

Egozy, E. (2016). Approaches to musical expression in Harmonix video games. In J. B. L. Smith, E. Chew, & G. Assayag (Eds.), *Mathemusical Conversations: Mathematics and Computation in Music Performance and Composition* (pp. 20–36). Singapore: Imperial College Press and World Scientific.

Engel, J., Holzer, M., Ruepp, O., & Sehnke, F. (2012). On computer integrated rationalized crossword puzzle manufacturing. In E. Kranakis, D. Krizanc, & F. Luccio (Eds.), Proceedings of the International Conference on Fun with Algorithms (pp. 131–141). Berlin, Heidelberg: Springer, Berlin Heidelberg.

Foote, J. (2000). *Automatic audio segmentation using a measure of audio novelty*. Proceedings of the IEEE International Conference on Multimedia & Expo (pp. 452–455). New York, NY: IEEE.

Ginsberg, M. L., Frank, M., Halpin, M. P., & Torrance, M. C. (1990). *Search lessons learned from crossword puzzles*. Proceedings of the National Conference on Artificial Intelligence (pp. 210–215). Boston, MA: AAAI Press.

Goto, M., Hashiguchi, H., Nishimura, T., & Oka, R. (2002). *RWC Music Database: Popular, classical, and jazz music databases*. Proceedings of the International Conference on Music Information Retrieval (pp. 287–288). Paris, France.

Goto, M., Hashiguchi, H., Nishimura, T., & Oka, R. (2003). *RWC Music Database: Music Genre Database and Musical Instrument Sound Database*. Proceedings of the International Conference on Music Information Retrieval (pp. 229–230). Baltimore, MD, USA.

Gottlieb, M. L. (1998). *Secrets of the MIT Mystery Hunt: An exploration of the theory underlying the construction of a multi-puzzle contest* (Bachelor's thesis). Cambridge, MA: Massachusetts Institute of Technology.

Greenberg, S., & Buxton, B. (2008). *Usability evaluation considered harmful (some of the time)*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 111–120). Florence, Italy: ACM.

Griffin, G., Kim, Y. E., & Turnbull, D. (2010). *Beat-sync-mash-coder: A web application for real-time creation of beat-synchronous music mashups*. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (pp. 437–440), Dallas, TX.

Griffiths, M. (2002). The educational benefits of videogames. *Education and Health, 20*, 47–51.

Hainsworth, S. (2006). Beat tracking and musical metre analysis. In A. Klapuri & M. Davy (Eds.), *Signal processing methods for music transcription* (pp. 101–29). New York, NY: Springer.

Hansen, K.F., Hiraga, R., Li, Z., & Wang, H. (2013). Music puzzle: An audio-based computer game that inspires to train listening abilities. In Advances in computer entertainment. *Lecture notes in computer science* (Vol. 8253, pp. 540–543). Cham, Switzerland: Springer International Publishing.

Hedges, S. A. (1978). Dice music in the eighteenth century. *Music & Letters, 59*, 180–187.

Hudson, N. J. (2011). Musical beauty and information compression: Complex to the ear but simple to the mind? *BMC Research Notes, 4*, 9.

Jordan, A., Scheftelowitsch, D., Lahni, J., Hartwecker, J., Kuchem, M., Walter-Huber, M., ..., Preuss, M., (2012). *BeatTheBeat: Music-based procedural content generation in a mobile game*. Proceedings of the IEEE Conference on Computational Intelligence and Games (pp. 320–327). Granada, Spain.

Krebs, F., Böck, S., & Widmer, G. (2013). *Rhythmic pattern modeling for beat and downbeat tracking in musical audio*. Proceedings of the International Society for Music Information Retrieval Conference (pp. 227–232). Curitiba, Brazil.

Lee, C., Lin, Y., Yao, Z., Lee, F., & Wu, J. (2015). *Automatic mashup creation by considering both vertical and horizontal mashabilities*. Proceedings of the International Society for Music Information Retrieval Conference (pp. 399–405). Málaga, Spain.

Lin, Y., Liu, I., Jang, J. R., & Wu, J. (2015). Audio musical dice game: A user-preference-aware medley generating system. *ACM Transactions on Multimedia Computing, Communications, and Applications*, *11*, 52:1–52:24.

Müllensiefen, D., Gingras, B., Musil, J. J., & Stewart, L. (2014). The musicality of non-musicians: An index for assessing musical sophistication in the general population. *PLoS ONE, 9*.

Müllensiefen, D., Gingras, B., Stewart, L., & Musil, J. J. (2013). *Goldsmiths musical sophistication index (gold-MSI) v1.0: Technical report and documentation [Revision 0.3]*. Technical report, Goldsmiths. University of London.

Müller, M. (2015). Music structure analysis. *Fundamentals of music processing: Audio, analysis, algorithms, applications* (pp. 167–236). Cham, Switzerland: Springer International Publishing.

Nierhaus, G. (2009). *Algorithmic Composition: Paradigms of Automated Music Generation*. Vienna, Austria: Springer-Verlag Wien.

Rentfrow, P. J., & Gosling, S. D. (2003). The do re mi's of everyday life: The structure and personality correlates of music preferences. *Journal of Personality and Social Psychology, 84*, 1236–1256.

Schmidhuber, J. (2009). Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. *Anticipatory behavior in adaptive learning systems* (pp. 48–76). Berlin: Springer.

Smith, J. B. L., Burgoyne, J. A., Fujinaga, I., De Roure, D., & Downie, J. S. (2011). *Design and creation of a large-scale database of structural annotations*. Proceedings of the International Society for Music Information Retrieval Conference (pp. 555–560). Miami, FL, USA.

Upham, F., & Farbood, M. (2013). *Coordination in musical tension and liking ratings of scrambled music*. Presented at the Society for Music Perception and Cognition Conference (pp. 148), Toronto, Canada.

Wyse, L., & Subramanian, S. (2013). The viability of the web browser as a computer music platform. *Computer Music Journal, 37*, 10–23.