CrossSong Puzzle: Generating and Unscrambling Music Mashups with Real-time Interactivity

Jordan B. L. Smith, Graham Percival, Jun Kato, Masataka Goto, Satoru Fukayama
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{ jordan.smith, graham-percival, jun.kato, m.goto, s.fukayama } @aist.go.jp

ABSTRACT

There is considerable interest in music-based games, as the popularity of Rock Band and others can attest, as well as puzzle games. However, these have rarely been combined. Most music-based games fall into the category of rhythm games, and in those games where music is incorporated into a puzzle-like challenge, music usually serves as either an accompaniment or reward. We set out to design a puzzle game where musical knowledge and analysis 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 truly integrates musical and logical reasoning. The game presents a player with a grid of tiles, each representing a mashup of measures from two different songs. The goal is to rearrange the tiles so that each row and column plays a continuous musical excerpt.

Automatically identifying a set of song fragments to fill a grid such that each tile contains an acceptable mashup is our primary technical hurdle. We propose an algorithm that analyses a corpus of music, searches the space of possible fragments, and selects an arrangement that maximizes the "mashability" of the resulting grid. This algorithm and the interaction design of the system are the main contributions.

1. INTRODUCTION

Why is listening to music enjoyable? One hypothesis is that a listener's pleasure derives from their ability to detect patterns in the music, thereby "compressing" it in their mind [1]. 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 [2]. Whether or not this is the only explanation, music shares an important trait with puzzles: pattern identification is central to the enjoyment of both. In the case of logic puzzles, such as sudoku, discovering patterns helps the solver to make deductions about how to complete the puzzle.

While being enjoyable for arguably similar reasons, there are few activities that target those with an interest in both music and puzzles. Devising a satisfying combination of active listening and puzzle-solving is a difficult task. Puzzles, including jigsaws and crosswords, are usually solved at a leisurely pace—interruptions are no hindrance—while music is defined by its happening in time, and interruptions or sudden changes in rhythm or playback can be disturbing. We have embraced the challenge of wedding these two forms together and have made a real-time puzzle game in which the solver listens to the audio "clues" without interruptions.

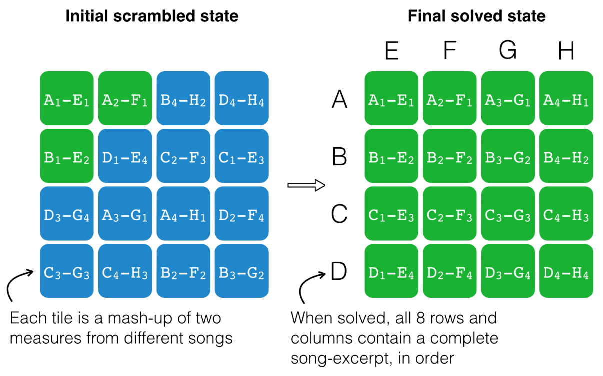


Figure 1: CrossSong puzzle overview. Green tiles indicate correct placement. The solver cannot see the labels and must deduce the correct order by listening to the tiles. Gameplay video:
`https://www.youtube.com/watch?v=I_i0HXPybKM`

The result is the CrossSong Puzzle (Figure 1). The puzzle consists of a 4x4 grid of tiles, where each row and column represents a four-measure excerpt of a song. Each tile thus represents one measure-long mashup of two songs. The solver is presented with a scrambled grid, and the object of the puzzle is to discover the correct arrangement of tiles by listening to them. Each excerpt has been time stretched to the same duration so that all beats match. Gameplay is continuous, with each tile playing one after the other with a constant tempo, to prevent the player from being distracted by the interruptions.

The puzzle resembles a musical version of a 4x4 sliding-tile puzzle, in which the goal is to reconstruct an image given similar constraints. However, it more strongly resembles a crossword puzzle in its construction. A crossword setter must find suitable words to fill a grid such that wherever two words cross, the same letter is used. Likewise, to make a pleasing CrossSong puzzle, we must find suitable song excerpts such that wherever two songs cross, a pleasing mashup is made. Discovering a set of excerpts where this is possible is a formidable but necessary challenge: if the mashups are dissonant or poorly matched rhythmically, the resulting discord will make gameplay tedious. The algorithm we developed for doing this, based on the work of [3], is one of our main contributions.

The other main contributions are the design of the puzzle itself and the interface used to solve it. Both were refined and tested iteratively, and the result is a puzzle that is challenging but accessible.

The rest of the paper is structured as follows. The next section reviews existing combinations of music and puzzles, as well as previous work in mashability estimation. Section 3 gives a formal overview of the proposed Cross-Song Puzzle design, including gameplay and implementation. Section 4 describes the algorithm that answers the technical challenge stated above. Section 5 discusses the iterative testing, design principles, and possible improvements to the system. We give concluding remarks in Section 6.

2. RELATED WORK

In this section, we first review prior effort on using music as part of interactive play, including examples in both pre and postcomputer age. Second, we review existing software for creating and estimating the quality of mashups.

2.1 Music as Part of Interactive Play

If we do not limit the scope to computer-aided puzzles, there is one tradition of musical puzzles that dates at least to the 15^{th} century: puzzle canons. The puzzle consists of a single monophonic melody, and the solver (usually a student of composition or other expert) must discover how to realize it as a canon. An early example of turning music-making into a game is the musical dice game of Western Europe, dating to the 1700s [4], in which random rolls of the dice were used to choose a selection of score fragments which were then performed for the amusement of the assembled.

There are many web- and smartphone-based games today which are based on music; however, a partial survey [5] suggests that the market is dominated by sound banks, multimedia players, instrument emulators, and music-creation apps like synthesizers and sequencers. Among the music-related puzzles we discovered, the link between the music and the puzzle mechanics were not very strong; in most cases, the logical reasoning is separate from the music, which serves more as a progress indicator or as a reward generated by the correct solution to the puzzle (e.g., Auditorium [1] , Chime [2] , Lumines [3]). Even when the music is deeply integrated into the puzzle structure, such as with FRACT OSC [4] , musical insight is not required to solve the challenges. Other related music-based games include the popular genre of rhythm games (e.g., Guitar Hero [5] , Hatsune Miku: Project DIVA [6] , Idolmaster [7])—but these are better described as physical challenges than as logic puzzles.

We would like to see a puzzle where the music is the *source* of information needed by the solver, and where careful listening is required. To our knowledge, the only predecessor with this feature is the puzzle game developed by Hansen et al. [6], 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, much like jigsaw pieces must be arranged in order to reconstruct an image. 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 a certain limitation, which ours aims to overcome. First, 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 will 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 support the view that rearranging parts of a piece of music at a sensible timescale does not necessarily disrupt one's enjoyment of the music [7].

2.2 Automatic Level Creation for Music Games

Creating levels for music games could be done with manual effort, but is cumbersome and makes it difficult to customize the gaming experience based on the users' needs. For instance, matching the audio clips to beat boundaries could be done with manual editing of the audio files, but a better approach is to generate levels based on rhythmic information extracted from the audio automatically. In this way, users can create levels based on their music libraries. Automatic methods of level creation have already been developed for music rhythm games such as Guitar Hero, Beat the Beat [8] and AudioSurf [8] .

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 mashup at multiple shifts in pitch. Many tools are capable of estimating beat locations to facilitate the creation of mashups, such as the Echo Nest Remix API [9] . Beat-Sync-Mash-Coder [9] computes this beat information and uses it to automatically synchronize the playback of two clips, but the portion of each song to use must be manually selected, and the system does not attempt to match the pitch of the clips. 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 mashups on their

[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://miku.sega.jp/arcade/en/
[7] http://idolmaster.jp/
[8] http://www.audio-surf.com/
[9] http://echonest.github.io/remix/
[10] http://mashup.mixedinkey.com/HowTo

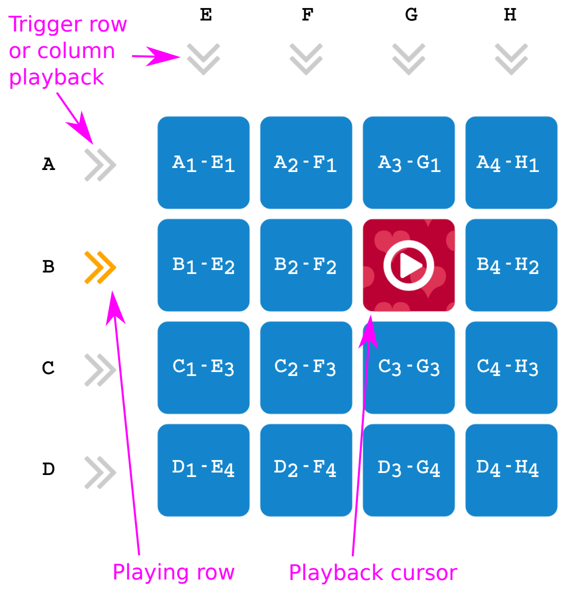
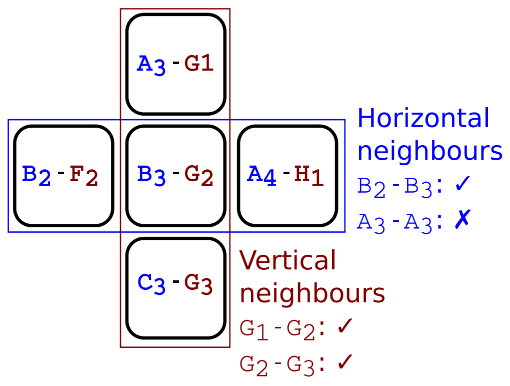
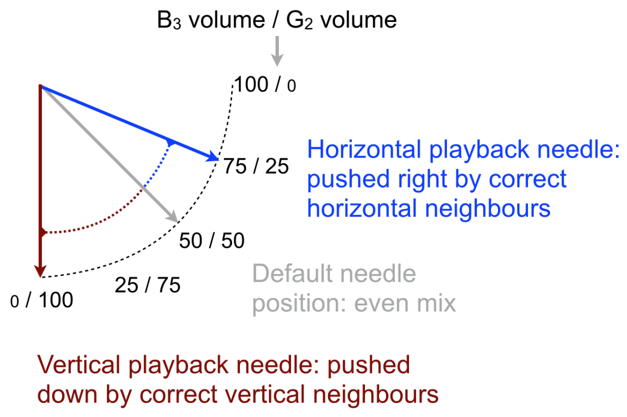


Figure 2: CrossSong Puzzle in its solved state, with labels added to each tile to illustrate the arrangement of music clips. Each tile contains a mashup of two clips; clip label X_i indicates the i^{th} measure of of song X. Solvers never see the tile labels, and begin with the tiles in random order.



(a) Illustration of relative cell correctness.



B3 volume / G2 volume
100 / 0
75 / 25
50 / 50
25 / 75
0 / 100
Horizontal playback needle: pushed right by correct horizontal neighbours
Default needle position: even mix
Vertical playback needle: pushed down by correct vertical neighbours

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

Figure 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.

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.

Among existing systems, AutoMashUpper [3] fulfills our requirements best. First, it performs beat, downbeat, and phrase-level boundary detection, since mashups between phrases that are intact and aligned downbeat-to-downbeat are understood to sound better. 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 [3]. In Section 4 we describe how the algorithm was adapted for our needs.

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. In the 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 excerpt X is 4 measures long; each of these measures, X_1–X_4, is associated with a different tile, and each tile is a mashup of measures from two songs. The solver begins the puzzle with the tiles arranged randomly and their task is to determine the correct order by listening to the tiles. Audio playback is continuous: the tiles are sounded in order from left to right, top to bottom, and the tile currently being played is highlighted. When the last column has finished playing, playback continues at the first row. All the tiles have the same duration and tempo, so even in the initial random configuration of tiles, the music has rhythmic coherence.

During gameplay, the solver can click on any two tiles to swap their position. They may also click on arrows outside the grid to choose which row or column to begin playing after the current one has ended. A link to a gameplay video is given in Figure 1. Solving a single puzzle takes roughly 10 minutes.

Normally, the two clips in each tile are played with equal loudness. However, as 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 3a. 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 3b. 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. It does not matter if B_3-G_2 is in the correct place in the 4x4 grid; this audio clue is based only on relative correctness.

3.1 Platform

We chose to implement 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 touchscreen, and even on a large-format touchscreen with multiple users (as pictured in Figure 6).

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 [10]. This allows the solving portion of the puzzle (as opposed to the generation phase) to scale to many users, as the server need only provide the html, css, javascript, and audio files to the user. The actual gameplay logic, as well as the audio mixing and scheduling, is performed on the local client computer.

Using a central server to generate and serve the audio has advantages and disadvantages. The main advantage is that we can perform audio analysis and generate puzzles using any language of our choice, rather than being restricted to javascript. Two disadvantages are that users are restricted to audio which is available on the server (i.e. they cannot use their own personal music collection), and if many users were attempting to create puzzles at the same time, the server could easily become overloaded. The latter problem is mitigated by caching all generated puzzles, so re-using an old puzzle has virtually no cost. Given that javascript audio-processing libraries are relatively new, we chose to use a central server.

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 8 song excerpts, each divisible into 4 equal-sized measures, such that, when arranged into a 4x4 grid, each combination of measures forms a good mashup.

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. The crossword generation problem, though seemingly straightforward relative to our task, has been researched for decades. It is a complex search problem that is NP-complete [11].

Our primary obstacle is the incredibly large 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 $8!/2 = 20,160$ ways of arranging them in the 4x4 grid. (The factor of 2 reduction recognizes that any arrangement and its transpose are equivalent.) Finally, each excerpt may be transposed up to 3 semitones upwards or downwards, increasing the space by a power of 7, approximately.

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

1. Compute audio features and detect phrase boundaries according to [3]
2. Compute mashability of all phrase-initial segments at all different delays. Retain mashability of optimal transposition of each.
3. Search loop:
   (a) Select one random excerpt from each song.
   (b) Find arrangement of these excerpts into grid with maximum mashability.
4. Repeat loop for pre-determined amount of time, and keep the best solution.
5. Process audio clips:
   (a) Apply time-stretching and pitch shift to match all excerpts using Rubberband library [12]
   (b) Match perceptual loudness of all excerpts using Replay Gain method [13]

4.1 Search optimizations

We first reduced the search space by restricting ourselves to excerpts that begin at one of the section boundaries estimated by AutoMashUpper. Doing so increases the odds that each excerpt will be an intact phrase of a song.

Our next optimization is to, for a pair of excerpts, only consider the transposition that gives the optimal mashability. This reduces the search space by a power of 7, but it can lead to problems: the final grid will require that all the clips be transposed to match each other, but these optimal transpositions can easily be infeasible. For example, suppose we choose clips A, B, E and F on the basis of their optimal mashability, disregarding the required transpositions. We may then match E_1 to A_1, F_1 to A_2, and B_1 to 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 measures (such as A_2 and F_1), but between full excerpts (such as A and F with the latter offset by one measure). This creates some mutual dependence in the mashability values. In the previous example, we can expect that B_2 and F_2 will match as long as B_1 and F_1 match. Assuming all the mashability values were 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.

4.2 Computation time and usability

Feature processing (step 1 in the list above) requires roughly 14 seconds to analyze each song (based on an average 3-minute song). Step 2, computing the mashability, takes roughly 0.5 seconds per pair of songs, or 14 seconds overall for an 8-song puzzle. For a given choice of 8 excerpts, all possible grid arrangements can be searched in roughly 0.03 seconds (step 3b). The remaining bottleneck is incredible number of random sets of excerpts, so we simply conduct a random search within a set time limit. In our tests, acceptable solutions were found in less than a minute of searching. Finally, the audio processing using Rubberband and Replay Gain takes about 10 seconds.

If the algorithm has access to the library beforehand, steps 1 and 2 of the algorithm can be executed in advance, in which case a good puzzle can be created in around a minute. Otherwise, an additional 2 minutes of pre-processing must take place.

Lastly, it should be noted that the algorithm makes many strong assumptions about the rhythmic regularity of the piece: constant tempo, constant 4/4 meter, and for the most part, phrases that are 2^n measures long. While these assumptions clearly do not apply to all music, they are prerequisites for our purpose. The user should be aware of this constraint and avoid selecting music in different time signatures. In the future, an automatic meter-detection step could be developed to quickly warn users of incompatible songs.

5. DESIGN DEVELOPMENT

A puzzle creator has two contradictory goals: first, to confront the solver with a problem that is very difficult to solve; and second, to ensure that the solver is eventually successful [14]. We iteratively tested a number of puzzle designs in order to strike a balance between posing no challenge and posing an insurmountable one. We also kept in mind some design criteria that are supported by the popular concept of "flow" [15], which seeks to explain why certain activities are more engaging than others. Namely, 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 and tested, including the pros and cons of each. 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 has handled.

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.

The puzzle was enjoyable to solve, 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 minutes without being certain of the relative position of any tiles, and was very discouraged.

Another problem is that we failed to realize that arranging the tiles in the transpose of the correct solution was logically sound, but not recognized by the system as correct.

Version 2: adding hints

Our second interface included strong visual hints to support the audio: the relative correctness of every tile was shown by displaying heart icons at the boundary with the correct neighbour (see Figure 4a). Also, to resolve the ambiguity of the solution, we added three fixed tiles in the top-left of the grid.

On the plus side, with a few fixed tiles to get started, solvers had an "in" to start the puzzle, and even solvers who were unfamiliar with the music could make progress. Unfortunately, the visual hints made progress far too rapid: once a few tiles had been placed in the correct order, the rest of the puzzle could be more easily as a visual packing problem, or simply by 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 adding any new information.

Version 3: refining visual hints

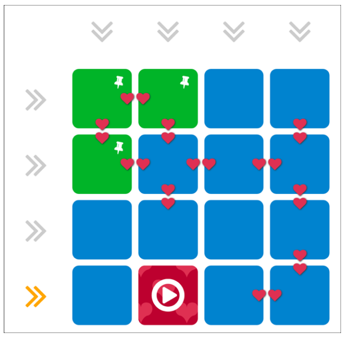
Our solution was to animate the background of the currently playing tile: we added a textured background that flows in the direction of the arrow in Figure 3b. For example, if no neighbours are correct, the background flows in a south-easterly direction; if both horizontal neighbours are correct during horizontal playback, the background flows east. 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 when the solver *listens* to the tile, so trial and error is too slow to be effective.

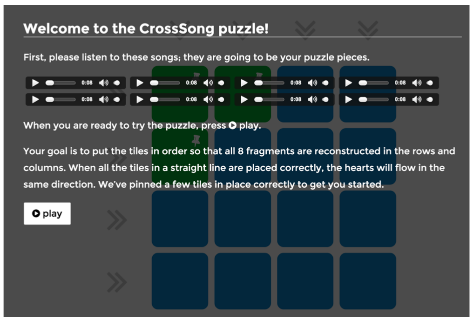
Those testing this version reported that the puzzle was still too difficult, for two reasons. First, mentally keeping track of the tiles was taxing, and it was easy to undo one's progress: for example, one might sort several similar tiles into a single row, but then forget which row it is, or accidentally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

Version 4: improving usability

We added two features to make the game more user-friendly. First, following the example of [6], we added a welcome screen (see 4b) where solvers were allowed to listen to each of the 8 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 4c). If all the tiles in a single row or column are placed in their


(a) Visual hints added to Version 2

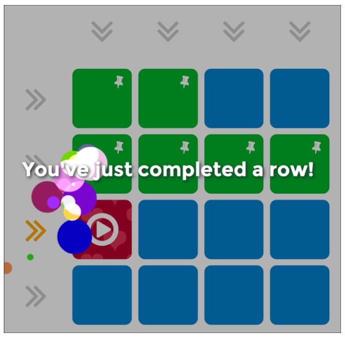

Welcome to the CrossSong puzzle!

First, please listen to these songs; they are going to be your puzzle pieces.

When you are ready to try the puzzle, press play.

Your goal is to put the tiles in order so that all 8 fragments are reconstructed in the rows and columns. When all the tiles in a straight line are placed correctly, the hearts will flow in the same direction. We've pinned a few tiles in place correctly to get you started.

(b) Welcome screen, added to Version 4



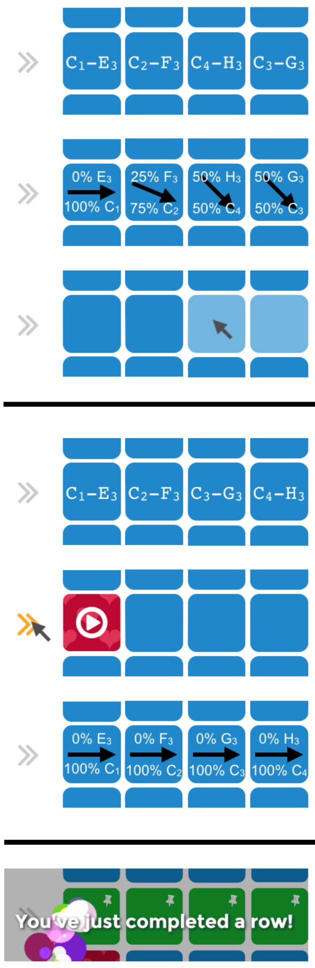
(c) Row confirmation screen, added to Version 4

Figure 4: Screenshots of development versions of CrossSong

correct position, a congratulatory message appears, and the tiles become fixed in place— but only after the full row (or column) is played, so that 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 are depicted in Figure 5.

This final version of the puzzle has most of the qualities we sought: it combines a need for careful listening with logical deduction, and although supported by visual hints, the visual hints do not dominate the puzzle-solving experience. The puzzle sets up a series of rewards (the row and column confirmations) that are achievable whether one is playing with one's favourite songs, or someone else's. Most of all, the game is fun. The game is available to play online [11] .

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



Game state: all tiles are in correct row, but order is incorrect.

User hears: clear, unmixed audio in first tile indicates first two tiles are correct.

User sees: user deduces which are incorrect, and clicks the remaining tiles to swap.

Game state: the row is now entirely correct.

User sees: after swap, user clicks arrow to listen to row again and check progress.

User hears: the audio is now all unmixed, so the user knows all tiles are correct. Success!

User sees: after listening to perfect row, they become fixed in place.

Figure 5: Depiction of a typical gameplay sequence. In the top part, the audio cues help the user identify which tiles arranged incorrect. In the middle part, the user listens to the new arrangement. The bottom part shows the visual feedback provided to the user.

Future versions

This section has mostly discussed the development of the core game mechanics, but there are other aspects of the game that could be refined. For example, in order to sustain one's engagement in CrossSong puzzles for more than a few levels, the layout of the initially fixed tiles should change for the sake of variety. Experienced solvers may wish to be able to turn off certain aids, such as the ability to pre-audition the excerpts, or to have correct rows fixed in place. Difficulty can also be increased by creating a larger puzzle; it is trivial to modify our algorithm to generate 8x8 puzzles.

One alternative version that we have implemented is the "multiplayer" mode. Two solvers each choose 4 songs, with excerpts from one solver's songs placed in the rows, and the others in the columns. (This constraint actually reduces the search space slightly for the algorithm in Section 4, reducing the computation time of step 3(b) from 30 ms to roughly 0.8 ms per iteration.) The solvers then work on the puzzle cooperatively on a large screen device (Figure 6).

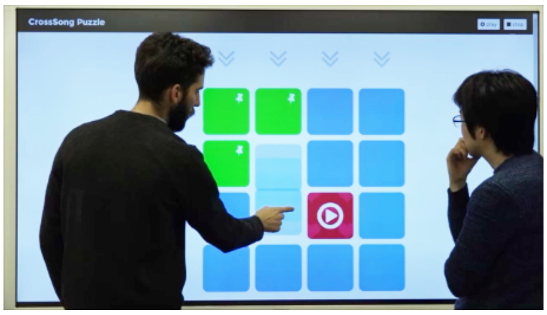


Figure 6: CrossSong Puzzle with two users working cooperatively.

6. 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, in real time, a set of original mashups.

The design was iteratively refined to focus the solver on the musical rather than the visual content, and to provide them with enough confirmation to make this task feasible. We would like to test the system on a larger scale to determine what parameter settings are preferred by a larger set of people. By tracking how fast each puzzle is solved, and the strategies used to solve them, we could refine the design so that the puzzle is rarely solved too quickly or too slowly. Both are outcomes that may reduce the enjoyability of the game.

The algorithm presented in Section 4 could be improved in several ways. For example, in pop songs, most sections are repetitions of other sections; if we detected these repetitions, we could ignore redundant sections and further reduce the search space. Second, the search space could be traversed more efficiently using probabilistic methods such as simulated annealing. Mashability could also be arbitrarily increased by treating the excerpts with harmonic-percussive source separation: this way, we could attempt to pair the drums from one song with the harmonies of another, reducing the severity of any harmonic or rhythmic incompatibility. Testing the usefulness of these improvements, as well as conducting larger-scale user testing, remain our future work.

Acknowledgement

We would like to thank Matthew Davies for his original implementation of AutoMashUpper [3]. This work was supported in part by OngaCREST, CREST, JST.

7. REFERENCES

[1] J. Schmidhuber, “Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes,” in *Anticipatory Behavior in Adaptive Learning Systems*. Springer, 2009.

[2] N. J. Hudson, “Musical beauty and information compression: Complex to the ear but simple to the mind?” *BMC research notes*, vol. 4, no. 1, 2011.

[3] M. E. P. Davies, P. Hamel, K. Yoshii, and M. Goto, “AutoMashUpper: Automatic creation of multi-song music mashups,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, 2014.

[4] S. A. Hedges, “Dice music in the eighteenth century,” *Music & Letters*, 1978.

[5] G. Dubus, K. F. Hansen, and R. Bresin, “An overview of sound and music applications for Android available on the market,” in *9th Sound and Music Computing Conference, SMC 2012*, 2012.

[6] K. F. Hansen, R. Hiraga, Z. Li, and H. Wang, “Music puzzle: An audio-based computer game that inspires to train listening abilities,” in *Advances in Computer Entertainment*, ser. Lecture Notes in Comp. Sci. Springer International Publishing, 2013, vol. 8253, pp. 540–543.

[7] F. Upham and M. Farbood, “Coordination in musical tension and liking ratings of scrambled music,” in *Presented at the Society for Music Perception and Cognition Conference*, 2013, p. 148.

[8] A. Jordan, D. Scheftelowitsch, J. Lahni, J. Hartwecker, M. Kuchem, M. Walter-Huber, N. Vortmeier, T. Delbrugger, U. Guler, I. Vatolkin, and M. Preuss, “BeatTheBeat: Music-based procedural content generation in a mobile game,” in *Computational Intelligence and Games (CIG)*, 2012.

[9] G. Griffin, Y. E. Kim, and D. Turnbull, “Beat-sync-mash-coder: A web application for real-time creation of beat-synchronous music mashups,” in *Acoustics Speech and Signal Processing (ICASSP)*, 2010.

[10] L. Wyse and S. Subramanian, “The viability of the web browser as a computer music platform,” *Computer Music Journal*, vol. 37, no. 4, 2013.

[11] M. L. Ginsberg, M. Frank, M. P. Halpin, and M. C. Torrance, “Search lessons learned from crossword puzzles,” in *Proc. of the Eighth National Conference on Artificial Intelligence - Volume 1*. AAAI Press, 1990.

[12] C. Cannam, “Rubber band library,” http://break-fastquay.com/rubberband.

[13] D. J. Robinson, “Perceptual model for assessment of coded audio.” Ph.D. dissertation, University of Essex, 2002.

[14] M. L. Gottlieb, “Secrets of the MIT Mystery Hunt: An exploration of the theory underlying the construction of a multi-puzzle contest,” 1998, Bachelor’s thesis.

[15] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*. New York, NY, USA: Harper and Row, 1990.