

# beat

music is everywhere™  
A High-Level Concept Document  
Andy Sage – March 2009

---

(Note: To avoid the many rhetoric problems that come from temporal confusion, this document will refer to *beat* in the present tense. “Current-generation” still refers to the games of 2009, though.)

## Overview

*beat*'s goal is to use various elements of the “Web 2.0 philosophy” to make the music game into less of a game and more of a fluid way to experience music. The basic premise of *beat* is that players should be able to choose exactly what music they want to play along to, how they want to play along to it, and where all of this takes place. Several things conspire to make this possible.

First is *beat*'s **platform independence**. In the current generation, all of the content for a game on a certain platform is encapsulated within that platform. *beat*, however, encourages players to experience it wherever and however they like. To this end, versions of *beat* exist for every major console, handheld, and mobile device in 2020, but the same content can be enjoyed on all of them! This is made possible by *beat*'s revolutionary approach to content management. When a user purchases a *beat* song module, that user's right to play that song extends to all *beat* games on all platforms. (See below for how gameplay works in this model.)

Supporting this platform independence is *beat*'s **distribution model**. Traditionally, music games come packaged with a number of “on-disc” songs that all users receive. Their flexibility comes from a diverse selection of downloadable content (DLC) that each player can pick and choose from. *beat* takes this a step further – *all* songs featured in *beat* can be chosen by the player from the very start. In a way, every song is “DLC”. Purchasing *beat* for a specific platform entitles the player to a certain number of prepaid song modules, to be selected from *beat*'s constantly-growing library of music. After these prepaid songs, additional modules can be purchased like traditional DLC, and delivered to any *beat* platform at any time. The platforms of 2020 are more than well-equipped enough to handle this kind of content delivery; omnipresent 3G data networks let any mobile device access the *beat* library at will, for example.

The most revolutionary aspect of *beat* is its modular approach to **gameplay**, which is what makes platform independence possible. Like most contemporary music games, the basic premise of *beat* is that the player is playing along to a specific music track as accurately as possible. But in contrast to traditional music games, *beat* has literally *no restrictions on what parts of the song are followed by the player*. Part of *beat*'s philosophy is that users should not be tied to a specific peripheral or platform to enjoy their favorite instrument. To make this possible, every version of *beat* features many different gameplay interfaces. While each of these is tied to a specific peripheral (i.e. a guitar

controller) or other mode of input (i.e. a touch-screen with a stylus), instruments are not tied to a specific interface – so a player on a home console can choose between a guitar controller and their console’s normal controller to play a song’s guitar part, for example. (Some restrictions will still be in place; using the guitar interface for a song’s drum part, for example, makes no sense.) Each *beat* song module will support at least two official interface charts for every track in the song, including vocals. For home consoles, this will usually be one for a specific peripheral and one for a normal controller; for handhelds and mobile devices, it’ll be two very different ways to use the device’s normal input to interact with the song. And, of course, *beat* has robust multiplayer modes – up to four different players, local or networked, can play together on songs that all players share, on any combination of instruments.

Why specify “official” charts? Another of *beat*’s notable features is its integrated support for **user-created content**, in the form of custom gameplay charts for official modules, using any compatible interface the user wishes. As stated above, every first-party *beat* module includes interfaces for each of the tracks included with the song. But what if a player is dissatisfied with a certain interface’s chart, or wants to play along on a peripheral not included in the original module? The console versions of *beat* provide in-game chart creation for any of its interfaces, including interfaces that don’t have to be tied to a specific instrument track. (This provides support for dance pad peripherals, for example, on electronica or dance songs.) A finite number of each user’s custom charts can be optionally hosted on *beat*’s central servers, where other users can download and rate them. The highest-rated charts will appear at the top of each song module’s “User Content” section, and each week, the *beat* development staff will select the best few for special recognition. Custom charts even fit in multiplayer modes, as the data needed to display custom charts is minimal, and can be sent to all players as required. This user-centric approach ensures that every song is enjoyed to its fullest by the entire player base.

While *beat* strives for platform-independence for purposes of content accessibility, each platform will offer unique takes on **other aspects** of the game (and, of course, each platform’s interfaces will vary greatly). Home console versions of *beat*, for example, might offer customizable avatars that can be improved and upgraded through a robust career mode tailored to that player’s track library. Consoles or handhelds that provide console-tied avatars could implement those in the game as well. Perhaps some modules could provide the song’s official music video. The game’s modular nature makes it possible to include many such song-specific elements in a module, such as avatar animations, instrument samples or effects to be used with appropriate interfaces, and bonus content like band trivia or artwork. (This document deals mostly with high-level concepts, so non-gameplay specifics like this are intentionally left vague. Developers for specific platforms will be given freedom to explore different ways to interpret the parts of each module.)

*beat* is a music experience unlike any other. If its scope sounds too grand to be possible, consider its chief design concepts: **modularity** and **accessibility**. Every aspect of *beat* is modular: every song is a self-contained unit that can be

interpreted in any number of ways, each of which is also its own module. The flexibility this gives a developer in distributing content, both songs and ways to play them, is immense. And because of this flexibility with user inputs, *beat* isn't tied to a specific genre of music – hip-hop can be enjoyed next to classic rock, or a player might segue from Metallica to Daft Punk. Combined with the user's complete control over their library, gone are the days of "I've never heard of any of these", or "I don't like this style of music". Never has a music game been so accessible! *beat*'s goal is to eventually become the norm for music distribution, and between its platform independent nature, the cross-demographic appeal of its wide variety of gameplay, and the replayability provided by user content, this is a goal that it can definitely achieve.

### **In-Depth Spotlight: Content Access for Handhelds**

While *beat* is unlikely to breach the inter-company gap and allow players on one console to play with those on another, it does something previously unheard of – your purchased content can be used on any version of the game, console or handheld. And while consoles' integrated hard drives can hold as many songs as you like, many handhelds and mobile devices still don't have the storage needed for your entire *beat* library. The solution? The 3G network. When using your 3G-enabled handheld or mobile device, *beat* will store only what it needs to locally, and will stream your chosen song straight to your device upon selecting it. Buffering will ensure uninterrupted gameplay, and frequently played songs will be cached to the device's internal or removable storage to make absolutely sure. (Songs can also be so cached manually.)

### **In-Depth Spotlight: More Content Distribution**

A developing trend in 2009 is the ability to purchase and download music onto an MP3 player from anywhere, without going through the hassle of transferring it from a computer. In 2020 this is the norm, and *beat* will take full advantage of every content distribution vector that it can. Console versions of *beat* can clearly access the song library through their integrated Internet connections, but the real potential here is for mobile devices. For example, let's say a band releases a new album, and all of its songs are charted for inclusion in the *beat* library. Flyers for the album's tour might include the 2020 equivalent of a QR code ([http://en.wikipedia.org/wiki/QR\\_Code](http://en.wikipedia.org/wiki/QR_Code)) or RFID tag ([http://en.wikipedia.org/wiki/Radio-frequency\\_identification](http://en.wikipedia.org/wiki/Radio-frequency_identification)) that, when scanned by *beat* on a supported mobile device, download a full song or short preview to the user's *beat* library for free. This would be fantastic promotion, and because of the nature of the *beat* library, the sample could be played on the user's console version of *beat* as well. If *beat* was truly integrated with the industry, the album itself could even contain each song's *beat* module, letting users extract the modules directly to their *beat* library. When *beat* is considered as its own song distribution vector – for all types of music! – the possibilities are vast.

### **In-Depth Spotlight: Gameplay Interfaces**

The modular, flexible nature of *beat* means that many gameplay interfaces can be successfully developed and deployed. The following examples have been selected to demonstrate this flexibility, covering a wide variety of peripherals, consoles, and types of music gameplay. (These examples are mostly based on existing music games. The nature of *beat* is such that almost any method of interacting with the song can be successful.)

#### **Interface: “Beat”**

**Platforms:** All (video game consoles, handhelds, mobile devices)

**Music Track:** Any instrumental track

**Peripherals:** Guitar controller, drum controller, keyboard controller, MIDI device (drums, keyboard, etc.), console controller, handheld/device face buttons; controller must have multiple button inputs

**Summary:** Hit scrolling notes to a single music track.

**Comparisons:** *Guitar Hero*, *Rock Band*, *Keyboardmania*, *Pop'n Music*, *beatmania IIDX*, *Dance Dance Revolution*, *FreQuency/Amplitude*, etc.

**Description:** Beat is *beat*'s primary interface, and the one that most closely mirrors traditional music games' gameplay. Beat covers a long list of instruments, and is actually a series of individual interfaces that differ based on the peripheral used (though all are labeled as **Beat** when choosing an interface for your controller). All Beat interfaces feature a row of targets at the bottom of a scrolling note field. Players must use their peripheral to hit “gems” (notes) as they pass the targets in sync with the instrument being played.

**Beat (guitar)** [guitar peripheral] uses the gameplay immortalized by titles like *Guitar Hero* and *Rock Band*: the player holds a specific fret button(s) and strums to hit each gem. Chords are played by holding multiple frets, and sometimes frets must be held after the strum to generate a held note. This interface controls acoustic guitars, electric guitars, and bass guitars.



*Guitar Hero II*

**Beat (drums)** [drum peripheral / MIDI drumkit] is much like *Rock Band* as well – there are four targets, each corresponding to a specific drum pad, and a line that stretches across all targets to represent a bass drum pedal. The drum pads and pedal are struck to hit each gem. This interface controls drum kits and some drum machines.



*Rock Band (drums)*

**Beat (keyboard)** [keyboard peripheral / MIDI keyboard] mimics *Keyboardmania*, and is unavailable in local multiplayer due to the size of the target area (players can use Beat (default) to play keyboard or synth in multiplayer). The target area shows a 24-note keyboard, reflected by the peripheral; the gems scroll down to these keys and are hit by pressing the right key at the right time. This interface controls piano, synthesizer, and some samples.



*Keyboardmania*

**Beat (controller)** [console controller / handheld] is similar to *FreQuency/Amplitude* or *beatmania IIDX*. Players must use the face buttons and right shoulder button of their controller to match the input displayed on-screen. Four buttons are used; the left, top, and right face buttons each have their own targets, and the right shoulder button is a line like Beat (drums)'s bass pedal. Gems/lines scroll towards these targets

and are hit by pressing the right button at the right time. This interface can be used for **any** instrumental track, but is the default interface for most samples and miscellaneous instruments without a dedicated peripheral.

**Beat (mobile)** [mobile device] mimics Beat (controller), but is tailored to fit four buttons on a mobile device.



*Guitar Hero Mobile*

### **Interface: “Groove”**

**Platforms:** Video game consoles and handhelds; some mobile devices

**Music Track:** Any pitched instrumental track, vocals (lead/background)

**Peripherals:** Console controller, motion sensing controller, handheld/mobile device face buttons

**Summary:** Hold a direction and hit one or two buttons in time to the track.

**Comparisons:** *Gitaroo Man*

**Description:** Groove is *beat*'s re-imagining of *Gitaroo Man*'s “Attack” and “Harmony” phases. The Groove play field centers on a single, fixed arrow that represents the player. “Phrases” – thick, colored lines – approach the dot at various angles, each with a circle at the beginning to mark a button press. Each phrase is connected by a small line that leads the player to the next phrase. **For console controllers and handhelds with analog sticks**, the left analog stick moves the arrow, which is held in the direction of the phrase, and a face button is pressed and held for the duration of the phrase. **For motion sensing controllers**, tilting the controller moves the arrow, and the primary button is pressed and held for the duration of the phrase. **For handhelds and mobile devices with a D-pad**, the phrases only come from the cardinal directions and diagonals; the D-pad moves the arrow, and a button is pressed and held for the duration of the phrase. **For all peripherals**, a second button may be used on harder difficulty levels to increase complexity. The player is judged on how accurately their

arrow direction and held button duration matched the phrase's angle and length, respectively.



*Gitaroo Man*

### **Interface: “Jam”**

**Platforms:** Motion-sensing consoles, touch-screen handhelds

**Music Track:** All

**Peripherals:** Motion sensing controllers, touch-screens

**Summary:** Tap and drag your pointer as directed to the music.

**Comparisons:** *Ouendan*, *Elite Beat Agents*

**Description:** Jam is an exclusively single-player mode, an example of an interface that doesn't rely on separated master tracks. Much like *Ouendan* and *Elite Beat Agents*, Jam presents the player with various elements they have to tap (“notes”) or tap, drag, and follow on a path (“holds”). **For motion sensing controllers**, the player has a cursor moved by the controller; this cursor is moved over an element and the main button is pressed to interact with it. **For touch-screen handhelds**, the player simply taps a stylus on the screen for the same functionality. The elements will generally follow the overall melody or beat of the song, and the player is judged on how accurately they can tap the notes to the beat.



*Ouendan*

## **Interface: “Virtuoso”**

**Platforms:** Video game consoles; supported handhelds/mobile devices

**Music Track:** Vocals (lead or background)

**Peripherals:** Microphone (external or internal)

**Summary:** Sing at the correct pitch.

**Comparisons:** *Karaoke Revolution*, *Rock Band*

**Description:** Virtuoso plays just like *Karaoke Revolution* and *Rock Band*.

A pitch indicator bar and lyrics scroll horizontally across the screen. As they hit a target area, players sing into the microphone; their pitch is detected and compared to the charted pitch of the song’s vocals track.

Like *Rock Band*, this system supports phoneme detection for unpitched singing. A Virtuoso part can be played on its own or combined with any Beat or Groove instrumental track; if combined, the Virtuoso scrolling area will be smaller and located at the top of that Beat/Groove track. (Virtuoso is supported on those handhelds and mobile devices whose built-in microphones are of high enough quality to accurately detect pitch.)



*Rock Band (vocals)*

---

## **In-Depth Spotlight: Song Selection Walkthrough**

This section will give a step-by-step impression of a player’s song and interface selection.

1. Player powers on her new 3G-enabled Nintendo touchscreen handheld, the *TSi. beat’s* flash cartridge is already loaded; Player puts headphones on, as she’s on a bus and doesn’t want to bother everyone with her rocking out.
2. Player doesn’t feel like playing online at the moment, and picks “Solo” from the main menu. She just wants to play, too, and doesn’t care about the story mode, so she picks “Quickplay”.
3. After a very short 3G buffering screen, Player’s list of 96 songs loads. She sorts it by artist, jumps to “D”, and sees the entirety of Daft Punk’s *Discovery*, her favorite album; she bought it last week. She decides to jam to “Digital Love”, and taps the song to select it.

4. Player is faced with the Interface Select screen. On the left side is a list of stylized icons that say “Beat”, “Groove”, “Jam”, and “Virtuoso”, the four interfaces supported by this song on this handheld. Feeling in the mood for some stylus action (and not really wanting to Virtuoso on a bus), Player selects “Jam”.
5. The middle of the screen displays a list of the charts available for Player to Jam to. Right now, she’s only got the four official charts: Easy, Medium, Hard, and Expert, all labeled with a star to show their authenticity and authored by “*beat*”. Player suddenly remembers her friend Player 2 bragging about his Digital Love Jam chart, and clicks the small “More Charts...” button at the bottom of the screen.
6. A second of 3G buffering later, Player sees a list of Jam charts for Digital Love. There are no official DLC charts for it yet, so the top entries are simply the best rated user-made entries. Lo and behold, Player spies “Digital Overload”, authored by “Player 2”, sitting at the third highest ranking. She taps it, gives its statistics a quick look, and taps “Download”. Chart data is small, so a few seconds later she’s back on the previous screen with the new chart ready to go.
7. Player taps “Start!”. After about 5 seconds of “Buffering...”, the familiar refrains of Daft Punk wash over her ears. Player uses her stylus to tap the patterns that start appearing; about thirty seconds in, she fails, and shouts a loud expletive at Player 2’s expense. After everyone stops staring at her, she returns sheepishly to *beat* and decides to play some songs on Groove for a while instead.

## **Conclusion**

*beat* represents not so much a game as it does a revolution in the way people see content delivery, music gaming as a platform for music enjoyment, and modular game design. This document intentionally leaves many specifics of *beat*’s gameplay vague, because that’s its greatest strength: adaptability. When viewed as a platform upon which gameplay can be constructed, instead of a rigid structure in itself, *beat* gives developers the freedom to do whatever they want with the game, on any platform they choose. In a way, *beat*’s true goal is the merger of the music industry and games industry, so that consumers will no longer see music as just music. Music is something people enjoy, and *beat* simply embodies new ways for this to happen. The technology available in 2020, as well as 10 more years of growth and evolution for music gaming, are what make this possible.

The ideas outlined in this document aren’t perfect. In particular, console developers are unlikely to agree to the distribution ideals of platform independence. (“What? They didn’t pay me directly for this content? No way!”) Potential solutions include a monthly *beat* licensing fee paid directly to the console companies to enable the use of a *beat* library on their system. But music games aren’t going anywhere; by 2020, song licensing issues will be much better-addressed than they are now.

I see *beat* as not only the evolution of music video games, but also as an example of how games will be handled after the Web 2.0 revolution. When the line is so thoroughly blurred between gaming consoles and PCs, for example, or between mobile phones, handheld games, and PDAs, it's critical that designers ensure that their games are modular, for the flexibility needed in design. And when this blurring of distinctions results in games becoming a mainstream form of entertainment, as we're already seeing quite a lot in 2009, the games need to be accessible to everyone.

This is what *beat's* all about – because music should be everywhere.

Andy Sage  
College of Technology, Purdue University  
pk.sage@gmail.com