

Io

a small programming language

Io overview

simple prototype-based object model

all actions are messages

simple and consistent syntax

dynamic all messages are dynamic

code is data and runtime modifiable

concurrent all objects can be actors

actors use coroutines

futures supported

and... bundled with extensive official bindings

The language

no keywords

no statements (only expressions)

expressions are composed only of messages

supports lexically scoped blocks

objects can have multiple parents

Message Syntax

a b

a b(c)

a b(c, d)

Operators

expression	compiles to
$a * 2 * b$	$a *(2) *(b)$

Assignment

expression	compiles to
<code>a := 2</code>	<code>setSlot("a", 2)</code>
<code>a = 2</code>	<code>updateSlot("a", 2)</code>

This separation allows self to be implicit

Loops

`while(x < 10, ...)`

`for(i, 1, 10, ...)`

`loop(...)`

`10 repeatTimes(...)`

Conditions

`a := if(b == 1, c, d) // conditions are expressions`

`if(a == b) then(`

`...`

`) elseif(...) then(`

`...`

`)`

Enumeration

```
someList := list("a", 2.3, "foo")
```

```
someList foreach(i, v,
```

```
    writeln(i, " : ", v)
```

```
)
```

```
// foreach also works on Maps, Strings, Buffers, etc
```

Blocks and Methods

```
foo := method(a, a + b) // object scoped
```

```
foo := block(a, a + b) // lexically scoped
```

Scoping

no globals

variables are local by default

Expressions

a := people select(person, person age < 30)

names := people map(i, person, person name)

“Macro” Example

```
glChunk := method(  
    glPushMatrix  
    sender doMessage(thisMessage argAt(0))  
    glPopMatrix  
)  
  
glChunk(glTranslated(1,2,3); glRectd(0,0,100,100))
```

Objects

```
Account := Object clone do(  
  balance := 0  
  deposit := method(amount,  
    balance = balance + amount  
  )  
)
```

Example

```
account := Account clone
```

```
account deposit(10.00)
```

```
writeln("balance:", account balance)
```

Everything is an Object

Number double := method(self * 2)

100 double

==> 200

Introspection

Number double := method(self * 2)

Number getSlot("double") code

==> "method(self *(2))"

Concurrency

url := URL with(“http://www.google.com”)

url fetch	// sync message
f := url @fetch	// future message
url @@fetch	// async message

Futures auto-detect deadlocks

io

Date (high precision, supports dates < 1970)

Duration

List

ImmutableSequence (Strings/Symbols)

Sequence (Buffers)

Map

WeakLink

server bindings

SGMLParser (supports XML and HTML)

Socket (async, libevent, supports async DNS)

Transparent Distributed Objects

Vector (supports SIMD/altivec)

Regex

SQLite3

MD5

Blowfish

CGI, URL

desktop bindings

OpenGL, GLU, GLUT

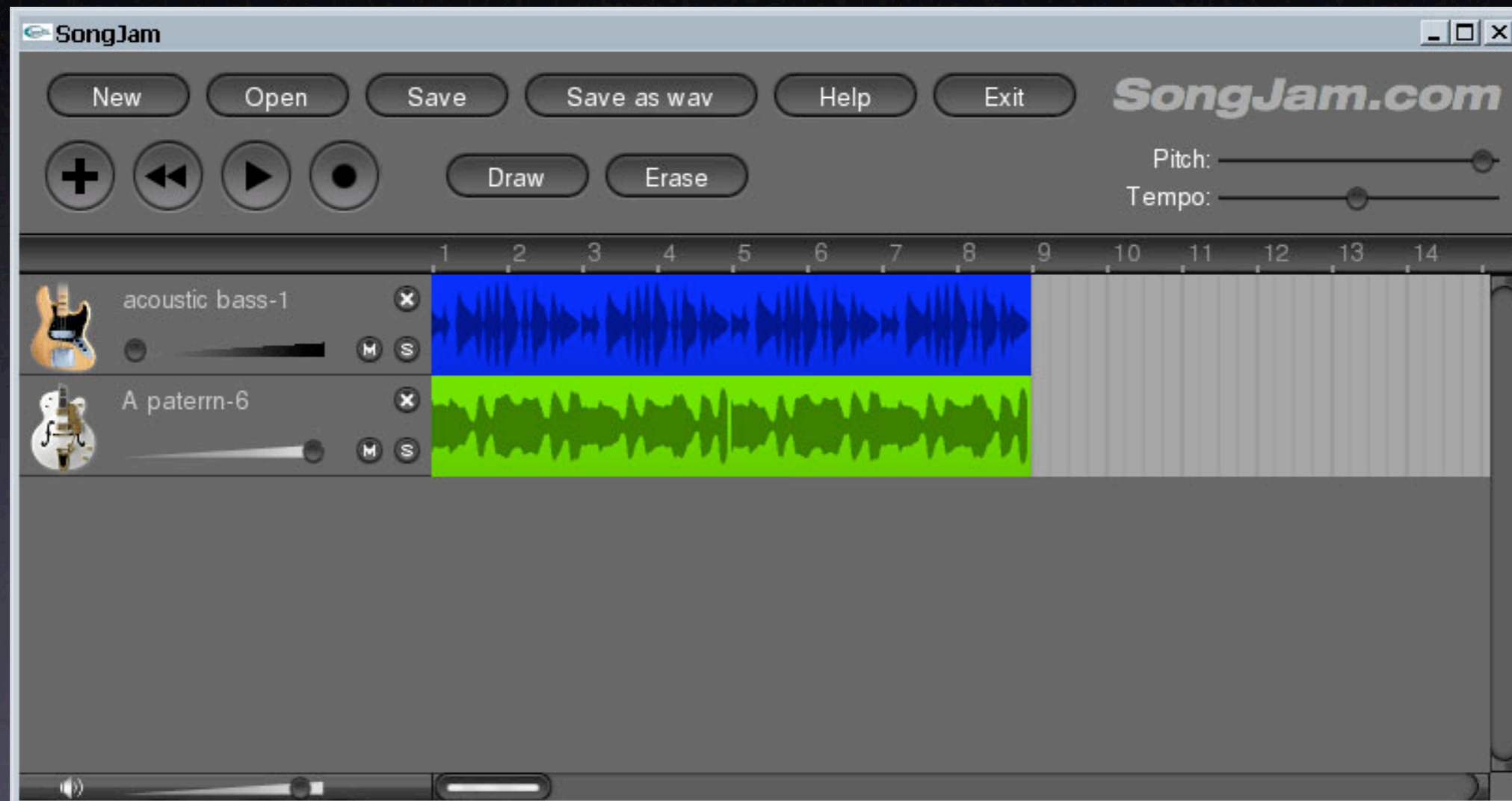
Audio (PortAudio)

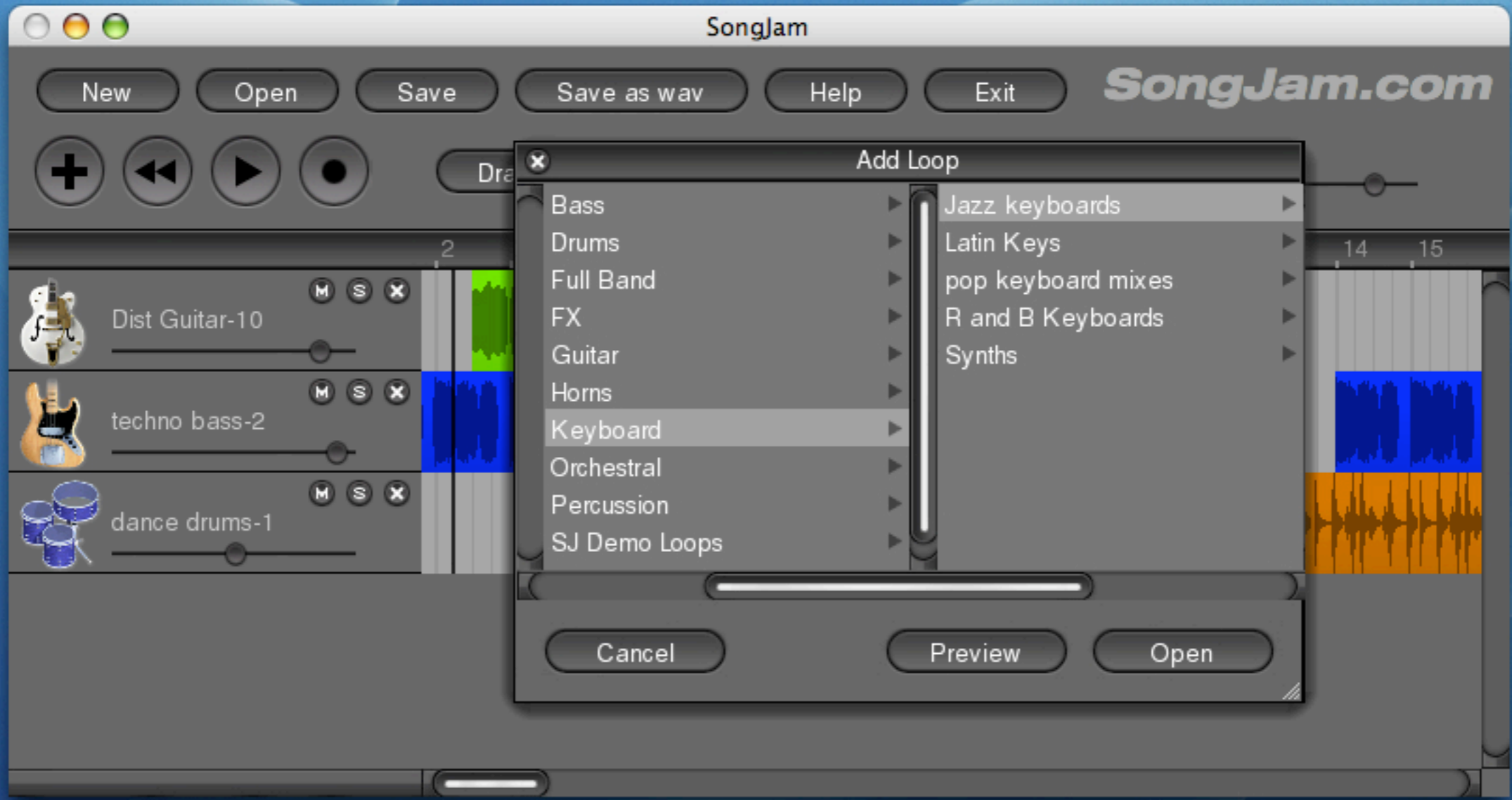
Font (FreeType, caches in texture)

Movie (ffmpeg)

Ion user interface toolkit

Ion example





Implementation

Garbage Collector

non-moving

tri-color

write-barrier

generational

Platforms

Unix OSX, Linux, *BSD, Irix

Windows Cygwin, Mingw, MSVC

Other Symbian, Syllable, Zeta

What's next?

Io 1.0 by end of 2005

incremental transparent persistence

improved transparent distributed objects

docs for Ion

bug tracker

revision control

official wiki

iolanguage.com