Turtle Tango (TT)

A line-drawing programming language that uses audio input to modulate graphical output
Contents
1. Introduction ........................................................................................................................................... 3
2. Language Tutorial ................................................................................................................................. 3
2.1. Starting State ...................................................................................................................................... 3
2.2. Hill (world) – A Simple Example ........................................................................................................... 4
2.3. Compiling and Running TT Files ........................................................................................................... 6
2.4. Addition Examples ................................................................................................................................. 7
3.1. Introduction ........................................................................................................................................ 12
3.2. Lexical Conventions ............................................................................................................................. 13
3.3. Scope .................................................................................................................................................. 13
3.4. Statements ......................................................................................................................................... 13
3.5. Statements ......................................................................................................................................... 15
3.6. Statements ......................................................................................................................................... 16
3.7. Routine Declarations ............................................................................................................................ 16
3.8. Built-in Routines ................................................................................................................................. 16
4. Project Plan ........................................................................................................................................... 17
4.1. Process .............................................................................................................................................. 17
4.2. Style Guide ....................................................................................................................................... 18
4.3. Timeline ........................................................................................................................................... 19
4.4. Software Development Environment .................................................................................................. 19
4.5. Project Log ....................................................................................................................................... 19
5. Architectural Design .............................................................................................................................. 20
5.1. Architecture ..................................................................................................................................... 20
5.2. Module Interfaces ............................................................................................................................... 20
6. Test Plan ............................................................................................................................................. 21
6.1. Example Translations ........................................................................................................................... 21
6.2. Test Suites ....................................................................................................................................... 21
6.3. Test Methodology ............................................................................................................................... 22
6.4. Test Automation ................................................................................................................................ 23
7. Lessons Learned .................................................................................................................................. 23
Appendix A: Turtle Tango Translator Code listing ................................................................................... 23
Appendix B: Translation Examples ............................................................................................................. 83
1. Introduction

Turtle Tango (TT) is a line-drawing language that uses audio input to modulate graphical output.

For a generation of programmers, Logo and turtle graphics provided a simple yet seductive introduction to writing code. By situating ourselves in the position of our friendly triangular Turtle, we envisioned the instructions required in order to induce the "motion" that created on-screen drawings. Given marching instructions and a pen pressed down, Turtle could draw for us whatever our sprouting CS-minds imagined.

TT nostalgically revisits a subset of turtle graphics in order to provide a learning vehicle for language and compiler design and implementation. To add novelty, deepen the project challenge, and inspire inventive graphical representations of music, TT augments Turtle's traditional trot by affording him the capacity to modify his strict response to marching command; the programmer can allow him some frivolity on the way to his destination by modifying his step in accordance to audio input. Turtle can tango to his favorite tunes.

Serving the role of the relative cursor, Turtle follows programming instructions and the modulations of a program's associated Tune to generate a PostScript representation of the specified dance in his world, the Disco. Although the applications for this language remain limited, the programmer retains the freedom to direct both Turtle’s stride and his Tune in order to create intriguing graphical representations of music. The technically-oriented musician can leverage TT to visualize his compositions while gaining insight in how to refine troublesome phrases. Thinking graphically may even inspire the desire to explore otherwise unconsidered musical techniques, leading to innovative creations unfathomed. Conversely, the technically-oriented graphic artist can leverage TT to incorporate structured axes of freedom into the regular patterns of line drawing. With the ability to augment his diagram with any Tune, he can explore the effects of various musical styles upon his artwork, thereby gaining insight into the relationship between his design, the music, and their symbiotic representation.

Perhaps most beneficially, TT maintains the spirit of Logo by empowering the novice programmer with the ability to type a dozen commands to produce an interesting drawing. The core syntax strays little from many imperative languages, and thus provides an appropriate introduction to future language learning. The addition of the musical analysis toolset provides an extra mode of seduction through which TT can lure the reluctant programmer. And the lure holds for the novice compiler programmer, as well; the graphical reward helps motivate the grind through the detailed work needed to construct and stabilize the translation system. By combining simplicity and familiarity while enabling comparative media creativity, Turtle Tango can help seduce the next generation of students so that in-demand CS-minds can thrive in the programming career dance.

2. Language Tutorial

2.1. Starting State

A meaningful TT program produces a single-page line diagram viewable in a PostScript viewer. The coordinate system ranges from the lower-left corner at (0, 24) and increases rightward and upward to the bounds of the page: (612, 742). At start, the Disco is white, and Turtle is in the lower-left corner at (0,24) pointing to the right 0 degrees with rate 1. The Pen is width 1, solid, and is black and down. The Tune is cued to sample 0 and is paused at volume
1. Turtle is ready to tango, but his path will only appear if, at the end of a sequence of instructions, his Pen is set to Up to mark the completion of this portion of his dance.

To create a PostScript drawing the programmer must compose at minimum a main routine, save it in a .tt file, and execute the TT translator with the .tt path and the desired .wav path. A simple example follows.

2.2. Hi! (world) – A Simple Example

The Hello World program in the Turtle Tango universe is a depiction of Turtle’s dance to the tune of a trumpet along a “Hi!” path. The code first declares a local variable `height` which we will use for the maximum height of our letters. Next Turtle is positioned with Pen Up to a good start location. We start the music in the Disco with the call to `play` and then have Turtle dance in the shape of an “H”, and “i”, and a “!” before lifting his pen. The code listing and resulting PostScript image follow.

```plaintext
/* hi.ttl */

routine main
{
  int height;
  height = 400;

  /* move to starting position */
  pu~;
  lt->45|;
  fd->250|;
  lt->45|;
  setLineWidth->3|;

  /* start up the tune in the Disco */
  play~;

  /* H */
  pd~;
  fd->height|;
  pu~;
  bk-> height/2 |;
  pd~;
  rt->90|;
  fd->100|;
  pu~;
  lt->90|;
  fd-> height/2 |;
  pd~;
  bk->height|;

  pu~;
  rt->90|;
  fd->50|;
  lt->90|;
```

/* I */
  pd~;
  fd-> height-150 |;
  pu~;
  fd->10|;
  setLineWidth-> 5|;
  pd~;
  fd->10|;
  pu~;

  setLineWidth-> 3|;
  rt->90|;
  fd->50|;
  lt->90|;
  fd->100|;

/* ! */
  pd~;
  rt->90|;
  fd->100|;
  rt->99|;
  fd-> height-100 |;
  rt->161|;
  fd-> height-100 |;
  pu~;
  bk-> height-100 |;
  setOrientation->270|;
  fd->10|;
  setLineWidth->15|;
  pd~;
  fd->30|;
  pu~;
}
2.3. Compiling and Running TT Files

Compiling a TT program requires a .tt program, a .wav audio file, and the tt compiler. Output should be directed to a specified PostScript output file. An example invocation from the Cygwin command prompt is:

```
./tt in.wav < in.tt > out.ps
```

The generated out.ps file can be opened in a PostScript viewer, or alternatively, printed.
2.4. Addition Examples

The first Hi! program can be extended to demonstrate the ease with which TT allows even simple drawing specification to transition toward creative abstraction. In the following code listing, the instructions to draw the “H” and the “i” have been factored into routines, and factored once more into a single “Hi” routine. Four Hi’s are then drawn along a diagonal from lower left to upper right with varying properties as Turtle continues to Tango to a trumpet. The once crisp lettering becomes a form reusable as a component in subsequent, free-form dances.

/* hi2.tt */

routine main
{
    int i;
    int height;
    height = 142;

    pu~;
    fd->5|
    lt->90|
    fd->5|

    play~;

    /* draw 5 Hi's in up-right diagonal with varying properties */
    for(i=1; i<5; i = i+1)
    {
        setRate-> i*2|
        setLineWidth->i|
        setDash-> i*2, i*2|
        setColor-> 10*i, 35*i, 45*i|
        setVolume-> 6-i |
        drawHi->height|;
        pu~;
        rt->90|
        fd->50|
        lt->90|
        ld-> height+10 |
    }
}

/* H */
routine drawH h
{
    pd~;
    fd->h|
    pu~;
    bk-> h/2 |
    pd~;
    rt->90|
    fd-> h/4 |
    pu~;
    lt->90|;
fd-> h/2 |;
pd-;
bk->h|;
}

/* i */
routine drawi h
{
  pd-;
  fd-> h * 5 / 8 |;
  pu-;
  fd-> 2 * (h/100) + 4 |;
  pd-;
  fd-> 2 * (h/100) + 4 |;
  pu-;
}

routine drawHi h
{
  drawH->h|;
  pu-;
  rt->90|;
  fd-> h/6 |;
  lt->90|;
  drawi->h|;
  bk-> h * 5/ 8 + 4 * (h/100) + 8|;
}
The first Hi! program also provides a useful starting point to demonstrate the last features of TT not yet explored. In the code listing below, for purpose of instruction, a global variable is declared to track the sample index, but it is recommended that a variable such as this one usually be declared locally. The Tune is cued to start at sample 1000. Out of the first 10000 samples, only groups of 42 are allowed to pass; the remaining samples are set to 0. The resulting PostScript image, also below, now resembles the form of a circuit diagram. Turtle Tango allows the technically-oriented artist to explore form quickly to encounter satisfying and sometimes surprising creations.
/* hi3.tt */

int sIndex; /* sample index */

routine main
{
    int i;
    int j;
    int height;
    sIndex=0;
    height = 400;

    /* cue the Tune to sample 1000 */
    cue->1000|;

    /* allow groups of 42 samples out of the first 10000 to pass unfiltered */
    for(i=1000; i < 10000; i=i+1)
    {
        sIndex = sIndex+1;
        if(sIndex == 42)
        {
            sIndex = 1;
            for(j=0; j<100; j=j+1)
            {
                setSample-> i+j, 0|;
            }
            i=i+j;
        }
    }

    /* move to starting position */
    setDiscoColor->0,190,0|;
    setColor->255,255,255|;
    pu~;
    lt->45|;
    fd->250|;
    lt->45|;
    setLineWidth->3|;

    /* start up the tune in the Disco */
    play~;

    /* H */
    pd~;
    fd->height|;
    pu~;
    bk-> height/2 |;
    pd~;
    rt->90|;
    fd->100|;
    pu~;
    lt->90|;
    fd-> height/2 |;
pd~;
bk->height|;

pu~;
rt->90|;
fd->50|;
lt->90|;

/* I */

pd~;
fd-> height-150 |;
pu~;
fd->10|;
setLineWidth-> 5|;
pd~;
fd->10|;
pu~;

setLineWidth-> 3|;
rt->90|;
fd->50|;
lt->90|;
fd->100|;

/* ! */

pd~;
rt->90|;
fd->100|;
rt->99|;
fd-> height-100 |;
rt->161|;
fd-> height-100 |;
pu~;
bk-> height-100 |;
setOrientation->270|;
fd->10|;
setLineWidth->15|;
pd~;
fd->30|;
pu~;
}

11

3.1. Introduction

A TT program consists of variable and routine declarations composed in a text file with a .tt extension. Control initiates in the main routine, which accepts a command-line file path argument pointing to .wav audio input. The output of compilation is a PostScript file with extension .ps depicting Turtle's dance in graphical form when viewed in a PostScript viewer.
3.2. Lexical Conventions

3.2.1. Tokens
The five classes of tokens are identifiers, keywords, constants, operators, and separators. Whitespace is ignored; TT is free-format.

3.2.2. Comments
The sequence ' /* ' begins a comment, which ends with the sequence ' */ '.

3.2.3. Identifiers
An identifier is a sequence of letters 'a'-'z' and 'A'-'Z' and digits '0'-'9' and '_'. Identifiers are case-sensitive, e.g. 'a' is not the same as 'A'.

3.2.4. Keywords
The following keywords are reserved:
- int, routine, return, if, else, for, while, print
- setDiscoColor, setColor, setDash, setLineWidth
- fd, bk, lt, rt, pu, pd
- setPosition, setOrientation, getOrientation, setRate
- cue, pause, play, stop, setVolume
- getTuneLength, getSample, setSample

3.2.5. Constants
Constants are integers consisting of a sequence of digits '0'-'9'.

3.3. Scope
Variable and routines are statically scoped.

Both global and local variable declarations are allowed; within a routine, local variables and formal parameters must have unique names. Local variables can only be accessed within the routine in which they are declared, and cannot be accessed before declared.

Routines must have globally unique names and cannot be declared within other routines. A routine can be called before it is declared.

3.4. Statements
Statements are executed in sequence unless otherwise noted.

3.4.1. Expression
Expressions are described in detail in section 5. An expression statement consists of an expression followed by ';'.

3.4.2. Compound
A compound statement allows several statements to be used in places in which one statement is expected, but more are needed. It consists of a '{' followed by zero or more statements and a closing '}'.

3.4.3. Conditional
Conditional expressions take one of two forms:

if (expression) statement
if (expression) statement else statement

The expression is first evaluated. Any resulting value other than zero is considered true, and the following statement is executed. If the expression evaluates to zero, in the first conditional form, the statement is skipped; in the second conditional form, the statement following the else is executed. else binds to the most recent if not associated with another else.

3.4.4. Loop

3.4.4.1. For

The for statement has one form:

    for (expression1; expression2; expression3) statement

expression1 specifies loop initialization. expression2 specifies a test, executed before each iteration, that evaluates to 0 to signal that the loop should exit. expression3 usually mutates a variable declared in expression1, so that expression2 eventually evaluates to 0. However, none of the three expressions is required. If expression2 is empty, the loop iterates until the break statement is executed.

3.4.4.2. While

The while statement has one form:

    while (expression) statement

The expression is evaluated once before each of zero or more iterations. If the expression evaluates to zero the loop terminates; otherwise, the statement is executed, followed again by the expression.

3.4.5. Return

A routine can terminate execution and return to its caller with the return statement, which has two forms:

    return;
    return expression;

When the expression is omitted, zero is returned.
3.5. Statements

Expressions are left-associative unless otherwise specified.

3.5.1. Constants
Integer constants are valid expressions.

3.5.2. Variables
A variable is accessed by specifying its identifier, which is a valid expression.

3.5.3. Unary Operators
The unary operator ' - ' negates its right-associated expression.

3.5.4. Binary Operators
The following binary operators are listed from highest to lowest precedence.

3.5.4.1. Multiplicative
'*' indicates: multiplication.
'/ ' indicates: division.

3.5.4.2. Additive
' + ' indicates: addition.
' - ' indicates: subtraction.

3.5.4.3. Relational and Equality
The relational and equality operators return 1 when true and 0 when false.
'< ' indicates: less than.
' > ' indicates: greater than.
' <= ' indicates: less than or equal.
' >= ' indicates: greater than or equal.
' == ' indicates: equal.
' != ' indicates: not equal.

3.5.5. Assignment
'= ' indicates: assignment. It is used to assign an integer variable or constant to a variable, and is right-associative.

3.5.6. Routine Calls
A routine without parameters is called by specifying its identifier, followed by a ' ~ '. A routine with parameters is called by specifying its identifier, the characters ' -> ', a comma-separated list of arguments, and the character ' | '. The number of the arguments in the comma-separated list must match the routine declaration.

    pd~;
    fd-> 10|;
3.6. Statements

A variable declaration has one form:

    int identifier;

The keyword int prefixes a variable declaration, and a semi-colon follows the identifier. Variables contain the value zero when first declared.

3.7. Routine Declarations

A routine declaration has the form:

    routine identifier arg1, arg2, ..., argN
    { variable declarations and statements }

The keyword routine is followed by an optional comma-separated list of arguments. A mandatory opening brace then begins a series of zero or more variable declarations and statements, followed by a closing brace.

All variables specified in the argument list are accessible as local variables; these variable names remain unique within the scope of the routine. The main routine transparently requires one argument, the FilePath used to initiate this program's Tune.

3.8. Built-in Routines

3.8.1. Disco

    setDiscoColor-> r,b,g|
    Sets the background color to the r,b,g value specified; r,b,g range from 0-255. The Disco starts with value 1,1,1.

3.8.2. Turtle

    fd-> x|, bk-> x|  
    Turtle moves forward or back positive x steps while drawing if Pen is down.
    lt-> d|, rt-> d|  
    Turtle turns left or right positive d degrees.
    setPosition-> x,y|  
    Turtle jumps to the position x,y without drawing. Turtle starts at position 0,24.
    getOrientation~  
    Retrieves Turtle's orientation in degrees. Turtle starts with orientation 0.
    setOrientation-> d|  
    Turtle turns to point in the direction of d degrees.
    setRate-> r|  
    Sets Turtle's rate of movement to r. Turtle starts with rate 1.
3.8.3. Pen

- *pu~*, *pd~*
  
  Sets the Pen Up, Down. Turtle starts with Pen Down.

- *setColor* > *r*, *b*, *g*
  
  Sets the Pen’s color to the *r*, *b*, *g* value specified; *r*, *b*, *g* range from 0-255. The Pen starts with value 0, 0, 0.

- *setDash* > *d*, *s*
  
  Sets the Pen to output strides of length *d* followed by space of size *s*. The Pen starts with value 1, 0 to specify a solid line.

- *setLineWidth* > *w*
  
  Sets the Pen’s Width to *w*. The Pen starts with value 1.

3.8.4. Tune

- *cue* > *s*
  
  Sets the sample index of the Tune. The Tune starts at sample 0.

- *pause~*
  
  Pauses the Tune in the Disco.

- *play~*
  
  Resumes play of the Tune in the Disco. The Tune is looped if play reaches the end.

- *setVolume* > *v*
  
  Sets the volume of the Tune to *v*. The Tune starts at volume 1.

- *getTuneLength~*
  
  Gets the number of samples in the Tune.

- *getSample* > *i*
  
  Gets the sample value at index *i*.

- *setSample* > *i*, *s*
  
  Sets the sample value at index *i* to *s*.

4. Project Plan

4.1. Process

The processes used for planning, specification, development, and testing each centered upon a bias for action. Planning was frequently set aside so that development experiments could assist in determining reasonable project scope. More broadly, though, planning proved critical for motivating the allocation of a sufficient quantity of time to establish a functional development environment; naiveté in the realms of Cygwin, Simple DirectMedia Layer, OCaml, and PostScript led to time-consuming confusion during efforts to isolate disconnects.

Specification required researching the PostScript language in order to determine the appropriate subset of instructions, as well as their proper orderings, needed to translate a TT program. Further inquiry into SDL and OCaml’s Bigarray module allowed for the finalization of the LRM.
The development process unfolded in three distinct phases. The first phase focused on the revision of the MicroC syntax to the one used by TT; since the languages are quite similar, the example implantation proved extremely valuable. Overcoming shift/reduce errors emerged as the primary concern during this phase. The second phase involved constructing the mechanisms to output sequences of PostScript instructions that resulted in intended drawings. Primary challenges in this phase included determining how to maintain state of the graphical properties, and of course, of the relative cursor, Turtle. The third and final phase of development required digging into the Audio library to uncover how to manipulate audio input. Printing the value of a single sample from audio input proved an overdue but inspirational milestone; accurately defining trigonometric calculations for Turtle’s movements led to some entertaining graphical manifestations.

Testing similarly proceeded in three phases and is discussed in greater detail in Section 6. A revised version of the test suite from MicroC established the validity of phase one development. The second phase required substantial visual inspection of the generated output PostScript files to confirm all operations rendered as intended. The third phase required significant tuning to ensure that audio modulation of drawings resulted in decipherable and interesting augmentations.

4.2. Style Guide

TT is free-form and thus does not explicitly demand a particular programming style. The recommended approach is to compose a TT program in a .tt text file. Global variables are defined at the top of the page. The main routine is defined first so that the entry point is immediately apparent to the reader. All other routines can be defined in any order, but the programmer should gather them into logical groupings. Indentation is optional but preferable for readability.

Readability is further improved by adding adequate spacing in routine calls. Generally, spacing should increase with the complexity of the arguments. For example:

```tt
foo->42;
```

is quite readable, but

```tt
foo->42, bar->80, 30, 4*5|, fubu~|;
```

could be more legible if written as:

```tt
foo-> 42, bar-> 80, 30, 4*5|, fubu~|;
```
or:

```tt
foo-> 42,
    bar-> 80, 30, 4*5|,
    fubu~|;
```

No further convention is specified, and the TT programmer retains discretion to format as desired. The intention of this minimal guidance is to allow the novice programmer to explore and develop personal preferences for modes of expression, and to allow the experienced programmer to apply practice used when writing in other languages in this domain.
4.3. Timeline

The table below contains the planned project timeline.

<table>
<thead>
<tr>
<th>Date</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>6/6/2012</td>
<td>Project Proposal</td>
</tr>
<tr>
<td>7/8/2012</td>
<td>PostScript &amp; Audio library research complete</td>
</tr>
<tr>
<td>7/15/2012</td>
<td>Development environment functional</td>
</tr>
<tr>
<td>7/22/2012</td>
<td>Final Scanner, Parser, Ast complete</td>
</tr>
<tr>
<td>7/29/2012</td>
<td>Hello World PostScript code generation</td>
</tr>
<tr>
<td>8/5/2012</td>
<td>PostScript code generation without audio</td>
</tr>
<tr>
<td>8/12/2012</td>
<td>PostScript code generation with audio</td>
</tr>
<tr>
<td>8/17/2012</td>
<td>Project Report</td>
</tr>
</tbody>
</table>

4.4. Software Development Environment

An unfortunate encounter with Boston's momentary monsoon season this July led to a transition of development environment midway through the project. Initial explorations used a 32-bit Windows XP notebook running a Windows OCaml compiler. The majority of the development has proceeded using a 64-bit Windows 7 machine running a Cygwin OCaml compiler. Installing a library that successfully loaded a .wav file in OCaml in this environment presented a surprisingly frustrating challenge, but once configured, development proceeded without interruptions. With minor adjustments, the Makefile included in the MicroC example proved very useful for keeping the build sequence simple and manageable. For reference, information about the audio library can be found here: [http://glcaml.sourceforge.net](http://glcaml.sourceforge.net).

4.5. Project Log

The table below contains the actual dates of project milestones.

<table>
<thead>
<tr>
<th>Date</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>5/26/2012</td>
<td>Project started – researching language ideas</td>
</tr>
<tr>
<td>6/6/2012</td>
<td>Project Proposal</td>
</tr>
<tr>
<td>6/17/2012</td>
<td>Initial grammar specified</td>
</tr>
<tr>
<td>6/24/2012</td>
<td>Initial Scanner, Parser, Ast complete</td>
</tr>
<tr>
<td>7/10/2012</td>
<td>PostScript and Audio library research complete</td>
</tr>
<tr>
<td>7/24/2012</td>
<td>Development environment functional</td>
</tr>
<tr>
<td>7/27/2012</td>
<td>Final Scanner, Parser, Ast complete</td>
</tr>
<tr>
<td>7/31/2012</td>
<td>Basic code generator complete; Testing phase 1 complete</td>
</tr>
<tr>
<td>8/2/2012</td>
<td>Hello World PostScript code generation</td>
</tr>
<tr>
<td>8/8/2012</td>
<td>PostScript code generation without audio; Testing phase 2 complete</td>
</tr>
<tr>
<td>8/15/2012</td>
<td>PostScript code generation with audio; Testing phase 3 complete</td>
</tr>
<tr>
<td>8/20/2012</td>
<td>Project Report</td>
</tr>
</tbody>
</table>
5. Architectural Design

5.1. Architecture

The Turtle Tango compiler and runtime environment adhere very closely to the architecture of the MicroC example provided. The only component removed is the interpreter; the TT system offers the capability solely to compile into a stack-based bytecode and execute in OCaml. The resulting output of a meaningful, compiled TT program is a set of PostScript instructions depicting an image when opened in a PostScript viewer.

The system consists of eight modules as depicted in the figure below: Lexical Analyzer (scanner.ml), Syntax/Semantic Analyzer (parser.ml), Abstract Syntax Tree (ast.ml), Bytecode Definition (bytecode.ml), Audio Library (sdl.ml, sdl_stub.c), Bytecode Generator (compile.ml), Runtime Environment (execute.ml), and User Executable (tt.ml).

5.2. Module Interfaces

The entry point of the compiler is the User Executable, which directs program control to the let statement marked main. Here, the lexer is invoked and interfaces with standard input by scanning TT code written in a .tt file. It divides this input into a set of valid tokens, and provides them as input to the parser’s program function. This function constructs a syntactically and semantically sensible representation of the written code by building the
abstract syntax tree’s declaration of `program` (Ast.program); if such a construction is not feasible, the parser interfaces with standard out by returning a Parse Error.

This organized list of program components is provided to the bytecode generator’s `Translate` function. This function translates the abstract syntax representation into a sequence of bytecode statements, using the bytecode definition; if such a translation is not possible, the bytecode generator interfaces with standard out by returning a useful error message, such as “undefined function foo”. If no error has occurred, the runtime environment accepts the bytecode statements and the path to the required input .wav file as input to its `execute_prog` function. With the assistance of an open source audio library that wraps Simple DirectMedia Layer for OCaml, the execution of these bytecode statements on the simulated stack produces a series of instructions describing the PostScript representation of the compiled TT program. These statements interface with standard out as a stream, and typically are redirected to a PostScript output file (such as out.ps).

Finally, the PostScript viewer provides a rewarding translation of PostScript statements into a visual image.

6. Test Plan

6.1. Example Translations

Provided in Appendix B are two examples of representative source programs and the associated generated PostScript. As a good introduction, first is the PostScript listing of the Hi! (world) example. The second example, similar to the program used in the second tutorial program, instructs Turtle to dance to his favorite trumpet, and his favorite step, Hi, while tracking his x and y position at a global level, and using his position to adjust the Pen color.

6.2. Test Suites

The following test programs were used regularly in an automated testing pattern. The tests in the left column test each of the basic imperative language constructs. The tests in the right column test each construct that impacts the Disco, Turtle, Pen, or Tune.

<table>
<thead>
<tr>
<th>test-arith1.tt</th>
<th>test-setPosition.tt</th>
</tr>
</thead>
<tbody>
<tr>
<td>test-arith2.tt</td>
<td>testSetColor.tt</td>
</tr>
<tr>
<td>test-fib.tt</td>
<td>test-setDiscoColor.tt</td>
</tr>
<tr>
<td>test-for1.tt</td>
<td>test-setLineWidth.tt</td>
</tr>
<tr>
<td>test-func1.tt</td>
<td>test-setDash.tt</td>
</tr>
<tr>
<td>test-func2.tt</td>
<td>test-setOrientation.tt</td>
</tr>
<tr>
<td>test-func3.tt</td>
<td>test-pd.tt</td>
</tr>
<tr>
<td>test-gcd.tt</td>
<td>test-pu.tt</td>
</tr>
<tr>
<td>test-global1.tt</td>
<td>test-fd.tt</td>
</tr>
<tr>
<td>test-hello.tt</td>
<td>test-bk.tt</td>
</tr>
<tr>
<td>test-if1.tt</td>
<td>test-rt.tt</td>
</tr>
<tr>
<td>test-if2.tt</td>
<td>test-lt.tt</td>
</tr>
<tr>
<td>test-if3.tt</td>
<td>test-setRate.tt</td>
</tr>
<tr>
<td>test-if4.tt</td>
<td>test-play.tt</td>
</tr>
<tr>
<td>test-ops1.tt</td>
<td>test-pause.tt</td>
</tr>
</tbody>
</table>
As a representative example, the contents of test-boundaries.tt and test-boundaries.out are listed below.

```c
/* test-boundaries.tt */
routine main
{
  pu~;
  fd->0|;
  lt->90|;
  fd->24|;
  pd~;
  fd->742|;
  rt->90|;
  fd->612|;
  rt->90|;
  fd->742|;
  rt->90|;
  fd->612|;
  pu~;
}

/* test-boundaries.out */
stroke
newpath
  0 24 moveto
stroke newpath
  0 24 moveto
  0. 0. rmoveto
  4.30775216872e-08 24. rmoveto
stroke
newpath
  0 48 moveto
  0.400001331813 741.999999999 rlineto
  612. 0.4 rlineto
  -0.60998668187 -742.000000001 rlineto
  -611.99999998 -0.610002196954 rlineto
stroke newpath
  0 45 moveto
showpage
```

### 6.3. Test Methodology

As discussed in section 4, the test plan involved three phases. The goal of the first phase was to stabilize the basic
imperative language without PostScript output by ensuring that all tests provided in the MicroC example test suite succeeded. These tests were included in each of the subsequent phases to ensure adequate regression coverage.

The goal of the second phase was to validate that all functions producing PostScript instructions were producing the intended images prior to introducing the musical component. Drawing a simple square served as an initial important test that was later extended to the bounds of the page to discover the precise dimensions of the Disco. Other critical verifications focused on the relative cursor and positioning system; the initial implementation led to buggy drawings as a result of mishandling Turtle’s movement when the Pen was up. Some dedicated stack-testing also became necessary due to initial inexact book-keeping of the stack pointer in execute.ml.

The goal of the third phase was to ensure that the audio modulations produced viable PostScript, rendered accurately, and produced interesting augmentations to the drawings. Although a set of tests were built for the automation process to ensure regression coverage, these tests relied primarily on inspection. The aim was to tune the system with proper constant coefficients for rates and volume so that the programmer would have an easy time using the audio samples of the input .wav file to create interesting Turtle Tangos.

6.4. Test Automation

All tests for each phase were included in the automated testing process. The architecture of this automation followed directly from the MicroC example. “Gold” .out files containing output inspected and verified as correct were compared with output generated during each round of tests to ensure that changes in one piece of the system did not break other components. The test suite was run frequently during development and refinement in an attempt to correlate fresh bugs with recent code changes.

7. Lessons Learned

The most enlightening lesson learned during the development of the TT system emerges when stepping back to review the set of choices made to construct the system. Although this effort leaned heavily on the MicroC example, the methodology to translate any set of characters in a text file into something entirely different, such as a mash-up of music and line-drawings, proves tractable. Furthermore, the infinite range of choices one can make in deciding how to translate one set of meaningful characters into another set that specifies programmatic instruction, reveals another dimension of freedom for the developer applying computer science. For example, this system could have translated TT code directly into PostScript without the bytecode intermediary; when starting the project, such a vision was difficult to perceive. Just as one programs as an act of creation when fluent in a language, one can extend this creative freedom beyond the scope of the language and into anything else. The challenge of square problems with round programming languages no longer must devolve into searches for messy compensatory libraries. The lesson is that the power to type in a text editor is the power to accomplish any programming endeavor.

Appendix A: Turtle Tango Translator Code listing

The following code listings are presented below: ast.ml, bytecode.ml, compile.ml, execute.ml, parser.mly, scanner.mll, sdl.ml, sdl_stub.c, tt.ml, testall.sh, Makefile.
**ast.ml**

type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq | Greater | Geq

type expr =
    | Literal of int
  | Id of string
  | Binop of expr * op * expr
  | Assign of string * expr
  | Call of string * expr list
  | Noexpr

type stmt =
    | Block of stmt list
  | Expr of expr
  | Return of expr
  | If of expr * stmt * stmt
  | For of expr * expr * expr * stmt
  | While of expr * stmt

type func_decl = {
  fname : string;
  formals : string list;
  locals : string list;
  body : stmt list;
}

type program = string list * func_decl list

**bytecode.ml**

type bstmt =
    Lit of int (* Push a literal *)
  | Drp (* Discard a value *)
  | Bin of Ast.op (* Perform arithmetic on top of stack *)
  | Lod of int (* Fetch global variable *)
  | Str of int (* Store global variable *)
  | Lfp of int (* Load frame pointer relative *)
  | Sfp of int (* Store frame pointer relative *)
  | Jsr of int (* Call function by absolute address *)
  | Ent of int (* Push FP, FP -> SP, SP += i *)
  | Rts of int (* Restore FP, SP, consume formals, push result *)
  | Beq of int (* Branch relative if top-of-stack is zero *)
  | Bne of int (* Branch relative if top-of-stack is non-zero *)
  | Bra of int (* Branch relative *)
  | Hlt (* Terminate *)

type prog = {
  num_globals : int; (* Number of global variables *)
  text : bstmt array; (* Code for all the functions *)
}
let string_of_stmt = function
  Lit(i) -> "Lit " ^ string_of_int i
| Drp        -> "Drp"
| Bin(Ast.Add) -> "Add"
| Bin(Ast.Sub) -> "Sub"
| Bin(Ast.Mult) -> "Mul"
| Bin(Ast.Div) -> "Div"
| Bin(Ast.Equal) -> "Eql"
| Bin(Ast.Neq) -> "Neq"
| Bin(Ast.Less) -> "Lt"
| Bin(Ast.Leq) -> "Leq"
| Bin(Ast.Greater) -> "Gt"
| Bin(Ast.Geq) -> "Geq"
| Lod(i) -> "Lod " ^ string_of_int i
| Str(i) -> "Str " ^ string_of_int i
| Lfp(i) -> "Lfp " ^ string_of_int i
| Sfp(i) -> "Sfp " ^ string_of_int i
| Jsr(i) -> "Jsr " ^ string_of_int i
| Ent(i) -> "Ent " ^ string_of_int i
| Rts(i) -> "Rts " ^ string_of_int i
| Bne(i) -> "Bne " ^ string_of_int i
| Beq(i) -> "Beq " ^ string_of_int i
| Bra(i) -> "Bra " ^ string_of_int i
| Hlt -> "Hlt"

let string_of_prog p =
  string_of_int p.num_globals ^ " global variables\n" ^
  let funca = Array.mapi
    (fun i s -> string_of_int i ^ " " ^ string_of_stmt s) p.text
  in String.concat "\n" (Array.to_list funca)

compile.ml

open Ast
open Bytecode

module StringMap = Map.Make(String)

(* Symbol table: Information about all the names in scope *)
type env = {
  function_index : int StringMap.t; (* Index for each function *)
  global_index   : int StringMap.t; (* "Address" for global variables *)
  local_index    : int StringMap.t; (* FP offset for args, locals *)
}

(* val enum : int -> 'a list -> (int * 'a) list *)
let rec enum stride n = function
  [] -> []
| hd::tl -> (n, hd) :: enum stride (n+stride) tl

(* val string_map_pairs StringMap 'a -> (int * 'a) list -> StringMap 'a *)
let string_map_pairs map pairs =  
  List.fold_left (fun m (i, n) -> StringMap.add n i m) map pairs

let big = 8 (* built-in global count *)

(** Translate a program in AST form into a bytecode program. Throw an  
    exception if something is wrong, e.g., a reference to an unknown  
    variable or function *)

let translate (globals, functions) =

  (* Allocate "addresses" for each global variable *)
  let built_in_globals = StringMap.add "orientation" 0 StringMap.empty in
  let built_in_globals = StringMap.add "rate" 1 built_in_globals in
  let built_in_globals = StringMap.add "tune_index" 2 built_in_globals in
  let built_in_globals = StringMap.add "is_playing" 3 built_in_globals in
  let built_in_globals = StringMap.add "volume" 4 built_in_globals in
  let built_in_globals = StringMap.add "is_drawing" 5 built_in_globals in
  let built_in_globals = StringMap.add "xpos" 6 built_in_globals in
  let built_in_globals = StringMap.add "ypos" 7 built_in_globals in
  let global_indexes = string_map_pairs built_in_globals (enum 1 big globals) in

  (* Assign indexes to function names; built-in "print" is special *)
  let built_in_functions = StringMap.add "print" (-1) StringMap.empty in
  let built_in_functions = StringMap.add "setPosition" (-2) built_in_functions in
  let built_in_functions = StringMap.add "setColor" (-3) built_in_functions in
  let built_in_functions = StringMap.add "setDiscoColor" (-4) built_in_functions in
  let built_in_functions = StringMap.add "setLineWidth" (-5) built_in_functions in
  let built_in_functions = StringMap.add "setDash" (-6) built_in_functions in
  let built_in_functions = StringMap.add "setOrientation" (-7) built_in_functions in
  let built_in_functions = StringMap.add "getOrientation" (-8) built_in_functions in
  let built_in_functions = StringMap.add "pd" (-10) built_in_functions in
  let built_in_functions = StringMap.add "pu" (-11) built_in_functions in
  let built_in_functions = StringMap.add "fd" (-12) built_in_functions in
  let built_in_functions = StringMap.add "bk" (-13) built_in_functions in
  let built_in_functions = StringMap.add "rt" (-14) built_in_functions in
  let built_in_functions = StringMap.add "lt" (-15) built_in_functions in
  let built_in_functions = StringMap.add "setRate" (-20) built_in_functions in
  let built_in_functions = StringMap.add "play" (-21) built_in_functions in
  let built_in_functions = StringMap.add "pause" (-22) built_in_functions in
  let built_in_functions = StringMap.add "cue" (-23) built_in_functions in
  let built_in_functions = StringMap.add "setVolume" (-24) built_in_functions in
  let built_in_functions = StringMap.add "getTuneLength" (-25) built_in_functions in
  let built_in_functions = StringMap.add "getSample" (-26) built_in_functions in
  let built_in_functions = StringMap.add "setSample" (-27) built_in_functions in
  let function_indexes = string_map_pairs built_in_functions
                          (enum 1 1 (List.map (fun f -> f.fname) functions)) in

  (* Translate a function in AST form into a list of bytecode statements *)
  let translate env fdecl =

  (* Bookkeeping: FP offsets for locals and arguments *)
  let num_formals = List.length fdecl.formals
  and num_locals = List.length fdecl.locals
  and local_offsets = enum 1 1 fdecl.locals
and formal_offsets = enum (-1) (-2) fdecl.formals in
let env = { env with local_index = string_map_pairs
  StringMap.empty (local_offsets @ formal_offsets) } in

let rec expr = function
  Literal i -> [Lit i]
| Id s ->
  (try [Lfp (StringMap.find s env.local_index)]
  with Not_found -> try [Lod (StringMap.find s env.global_index)]
  with Not_found -> raise (Failure ("undeclared variable " ^ s)))
| Binop (e1, op, e2) -> expr e1 @ expr e2 @ [Bin op]
| Assign (s, e) -> expr e @
  (try [Sfp (StringMap.find s env.local_index)]
  with Not_found -> try [Str (StringMap.find s env.global_index)]
  with Not_found -> raise (Failure ("undeclared variable " ^ s)))
| Call (fname, actuals) -> try
  (List.concat (List.map expr (List.rev actuals))) @
  [Jsr (StringMap.find fname env.function_index) ]
  with Not_found -> raise (Failure ("undefined function " ^ fname)))
| Noexpr -> []

in let rec stmt = function
  Block sl -> List.concat (List.map stmt sl)
| Expr e -> expr e @ [Drp]
| Return e -> expr e @ [Rts num_formals]
| If (p, t, f) -> let t' = stmt t and f' = stmt f in
  expr p @ [Beq(2 + List.length t')] @
  t' @ [Bra(1 + List.length f')] @ f'
| For (e1, e2, e3, b) ->
  stmt (Block([Expr(e1); While(e2, Block([b; Expr(e3)]))]))
| While (e, b) ->
  let b' = stmt b and e' = expr e in
  [Bra (l+ List.length b')] @ b' @ e' @
  [Bne (~(List.length b' + List.length e'))]

in [Ent num_locals] @  (* Entry: allocate space for locals *)
stmt (Block fdecl.body) @  (* Body *)
[Lit 0; Rts num_formals]   (* Default = return 0 *)

in let env = { function_index = function_indexes;
  global_index = global_indexes;
  local_index = StringMap.empty } in

(* Code executed to start the program: Jsr pu; Jsr main; halt *)
let entry_function = try (* initialize Disco, Pen at 0,0 *)
  [Jsr (-10) ; Jsr (StringMap.find "main" function_indexes); Hlt]
with Not_found -> raise (Failure ("no ""main"" function"))

in (* Compile the functions *)
let func_bodies = entry_function :: List.map (translate env) functions in

(* Calculate function entry points by adding their lengths *)
let (fun_offset_list, _) = List.fold_left
  (fun (l,i) f -> (i :: l, (i + List.length f))) ([],0) func_bodies in
let func_offset = Array.of_list (List.rev fun_offset_list) in

{ num_globals = List.length globals + big;
  (* Concatenate the compiled functions and replace the function
     indexes in Jsr statements with PC values *)
  text = Array.of_list (List.map (function
    Jsr i when i > 0 -> Jsr func_offset.(i)
  | _ as s -> s) (List.concat func_bodies))
}

execute.ml
open Sdl
open Audio
open Ast
open Bytecode

(* Stack layout just after "Ent":

   <-- SP
Local n
...
Local 0
Saved FP  <-- FP
Saved PC
Arg 0
...
Arg n *)

let execute_prog prog path =
  let stack = Array.make 131072 0
  and globals = Array.make prog.num_globals 0
  and tr = 0.0174532925 (* to_radian conversion *)
  and rc = 0.5 (* rate coefficient *)
  and vc = 0.01 in (* volume coefficient *)
  let a, tune = load_wav path in
  let tlength = Bigarray.Array1.dim tune in
  globals.(1) <- 1; (* initialize rate to 1 *)
  globals.(4) <- 1; (* initialize volume to 1 *)
  globals.(7) <- 24; (* initialize Turtle to lower left coordinate (0,24) *)

let tango dir dist =
  let position = Array.make 2 0.0 in (* local float position *)
  let rate = rc *. float_of_int globals.(1) in
  let radians = ((float_of_int globals.(0)) *. tr) in
  let dance = if globals.(5) > 0 then 1.0 else 0.0 in (* need pen down *)
  let steps = (* only dance to tune if playing *)
    (if globals.(3) > 0 && globals.(5) > 0 then int_of_float
      (ceil (float_of_int dist /. rate)) else 1) in
  let stride = (if globals.(3) > 0 && globals.(5) > 0 then rate


else (float_of_int dist)) in
position.(0) <- float_of_int globals.(6); position.(1) <- float_of_int globals.(7);

for i = 1 to steps do

if (globals.(2) >= tlength) then globals.(2) <- 0;

let x =
(-1.0 *. (sin radians)) *. dance *. (* dance modulation *)
(float_of_int (tune.{globals.(2)}) -
(if globals.(2) > 0 then
tune.(globals.(2)-1) else 0)) *. (float_of_int globals.(4)) *. vc
+. (* + stride *)
(dir *. stride *. (cos radians)) in

let y =
(cos radians) *. dance *. (* dance modulation *)
(float_of_int (tune.{globals.(2)}) -
(if globals.(2) > 0 then
tune.(globals.(2)-1) else 0)) *. (float_of_int globals.(4)) *. vc
+. (* + stride *)
(dir *. stride *. (sin radians)) in

print_endline (string_of_float x ^^ " "
string_of_float y ^
if globals.(5) > 0 then " rlineto " else " rmoveto "); (* pd|pu *)
globals.(2) <- globals.(2) + 1; (* update tune_index *)
position.(0) <- position.(0) +. x; (* update current position *)
position.(1) <- position.(1) +. y;

done;
globals.(6) <- int_of_float position.(0);
globals.(7) <- int_of_float position.(1);
in

let rec exec fp sp pc = match prog.text.(pc) with
| Lit i -> stack.(sp) <- i; exec fp (sp+1) (pc+1)
| Drp -> exec fp (sp-1) (pc+1)
| Bin op -> let op1 = stack.(sp-2) and op2 = stack.(sp-1) in
  stack.(sp-2) <- (let boolean i = if i then 1 else 0 in
  match op with
  Add -> op1 + op2
  | Sub -> op1 - op2
  | Mult -> op1 * op2
  | Div -> op1 / op2
  | Equal -> boolean (op1 = op2)
  | Neq -> boolean (op1 != op2)
  | Less -> boolean (op1 < op2)
  | Leq -> boolean (op1 <= op2)
  | Greater -> boolean (op1 > op2)
  | Geq -> boolean (op1 >= op2)) ;
exec fp (sp-1) (pc+1)
| Lod i -> stack.(sp) <- globals.(i); exec fp (sp+1) (pc+1)
| Str i -> globals.(i) <- stack.(sp-1); exec fp sp (pc+1)
| Lfp i -> stack.(sp) <- stack.(fp+i); exec fp (sp+1) (pc+1)
| Sfp i -> stack.(fp+i) <- stack.(sp-1); exec fp sp (pc+1)
| Jsr(-1) -> print_endline (string_of_int stack.(sp-1));
| exec fp sp (pc+1) (* print *)
| Jsr(-2) -> print_endline (string_of_int stack.(sp-1)^" "^string_of_int stack.(sp-2)^" moveto ");
| globals.(6) <- stack.(sp-1);
globals.(7) <- stack.(sp-2);
exec fp sp (pc+1) (* setPosition *)
| Jsr(-3) -> print_endline
| (string_of_float (float_of_int(stack.(sp-1))/255.0)^" "^string_of_float (float_of_int(stack.(sp-2))/255.0)^" "^string_of_float (float_of_int(stack.(sp-3))/255.0)^" "^"setrgbcolor "); (* setColor *)
| exec fp sp (pc+1)
| Jsr(-4) -> print_endline
| (string_of_float (float_of_int(stack.(sp-1))/255.0)^" "^string_of_float (float_of_int(stack.(sp-2))/255.0)^" "^string_of_float (float_of_int(stack.(sp-3))/255.0)^" "^"setrgbcolor clippath fill "); (* setDiscoColor *)
| exec fp sp (pc+1)
| Jsr(-5) -> print_endline (string_of_int stack.(sp-1)^" setlinewidth ");
| exec fp sp (pc+1) (* setLineWidth *)
| Jsr(-6) -> print_endline ("["^string_of_int stack.(sp-1)^" "^string_of_int stack.(sp-2)^" ] 0 setdash ");
| exec fp sp (pc+1) (* setDash *)
| Jsr(-7) -> globals.(0) <- stack.(sp-1); (* setOrientation @ g0 *)
| exec fp sp (pc+1)
| Jsr(-8) -> stack.(sp) <- globals.(0); (* getOrientation @ g0 *)
| exec fp (sp+1) (pc+1)
| Jsr(-10) -> print_endline ("stroke ");
| print_endline("newpath ");
| print_endline(string_of_int globals.(6)^" "^string_of_int globals.(7)^" "^moveto ");
| globals.(5) <- 1;
| exec fp (sp+1) (pc+1) (* pd *)
| Jsr(-11) -> print_endline ("stroke newpath ");
| print_endline(string_of_int globals.(6)^" "^string_of_int globals.(7)^" "^moveto ");
| globals.(5) <- 0;
| exec fp (sp+1) (pc+1) (* pu *)
| Jsr(-12) -> tango 1.0 stack.(sp-1); (* fd *)
| exec fp sp (pc+1)
| Jsr(-13) -> tango (-1.0) stack.(sp-1); (* bk *)
| exec fp sp (pc+1)
| Jsr(-14) -> globals.(0) <- (globals.(0) - stack.(sp-1)); (* rt *)
| exec fp sp (pc+1)
| Jsr(-15) -> globals.(0) <- (globals.(0) + stack.(sp-1)); (* lt *)
| exec fp sp (pc+1)
Jsr(-20) -> globals.(1) <- stack.(sp-1); (* setRate: rate @ g1 *)
  exec fp sp (pc+1)
Jsr(-21) -> globals.(3) <- 1; (* play: is_playing @ g3 *)
  exec fp (sp+1) (pc+1)
Jsr(-22) -> globals.(3) <- (-1); (* pause: is_playing @ g3 *)
  exec fp (sp+1) (pc+1)
Jsr(-23) -> globals.(2) <- stack.(sp-1); (* cue: tune_index @ g2 *)
  exec fp sp (pc+1)
Jsr(-24) -> globals.(4) <- stack.(sp-1); (* setVolume: volume @ g4 *)
  exec fp sp (pc+1)
Jsr(-25) -> stack.(sp) <- tlength; (* getTuneLength *)
  exec fp (sp+1) (pc+1)
Jsr(-26) -> stack.(sp) <- tune.{stack.(sp-1)}; (* getSample *)
  exec fp sp (pc+1)
Jsr(-27) -> tune.{stack.(sp-1)} <- stack.(sp-2); (* setSample *)
  exec fp sp (pc+1)
Jsr i -> stack.(sp) <- pc + 1; exec fp (sp+1) i
Ent i -> stack.(sp) <- fp; exec sp (sp+i+1) (pc+1)
Rts i -> let new_fp = stack.(fp) and new_pc = stack.(fp-1) in
  stack.(fp-1-1) <- stack.(sp-1); exec new_fp (fp-i) new_pc
Beq i -> exec fp (sp-1) (pc + if stack.(sp-1) = 0 then i else 1)
Bne i -> exec fp (sp-1) (pc + if stack.(sp-1) != 0 then i else 1)
Bra i -> exec fp sp (pc+i)
Hlt -> print_endline("showpage"); free_wav tune; ()

in exec 0 0 0

parser.mly

{% open Ast %}

%token SEMI LPAREN RPAREN LBRACE RBRACE COMMA
%token PLUS MINUS TIMES DIVIDE ASSIGN
%token EQ NEQ LT LEQ GT GEQ
%token RETURN IF ELSE FOR WHILE INT
%token ROUTINE WITH EXEC IEXEC
%token <int> LITERAL
%token <string> ID
%token EOF

%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE

%start program
%type <Ast.program> program

%%
program:
   /* nothing */ { [], [] }
| program vdecl { ($2 :: fst $1), snd $1 }
| program fdecl { fst $1, ($2 :: snd $1) }

fdecl:
   ROUTINE ID formals_opt LBRACE vdecl_list stmt_list RBRACE
   { { fname = $2;
       formals = $3;
       locals = List.rev $5;
       body = List.rev $6 } }

formals_opt:
   /* nothing */ { [] }
| formal_list { List.rev $1 }

formal_list:
   ID
   { [$1] }
| formal_list COMMA ID { $3 :: $1 }

vdecl_list:
   /* nothing */ { [] }
| vdecl_list vdecl { $2 :: $1 }

vdecl:
   INT ID SEMI { $2 }

stmt_list:
   /* nothing */ { [] }
| stmt_list stmt { $2 :: $1 }

stmt:
   expr SEMI { Expr($1) }
| RETURN expr SEMI { Return($2) }
| LBRACE stmt_list RBRACE { Block(List.rev $2) }
| IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5, Block([])) }
| IF LPAREN expr RPAREN stmt ELSE stmt { If($3, $5, $7) }
| FOR LPAREN expr_opt SEMI expr_opt SEMI expr_opt RPAREN stmt
   { For($3, $5, $7, $9) }
| WHILE LPAREN expr RPAREN stmt { While($3, $5) }

expr_opt:
   /* nothing */ { Noexpr }
| expr { $1 }

expr:
   LITERAL
   { Literal($1) }
| ID
   { Id($1) }
| expr PLUS expr { Binop($1, Add, $3) }
| expr MINUS expr { Binop($1, Sub, $3) }
| expr TIMES expr { Binop($1, Mult, $3) }
| expr DIVIDE expr { Binop($1, Div, $3) }
| expr EQ expr { Binop($1, Equal, $3) } |
| expr NEQ expr { Binop($1, Neq, $3) } |
| expr LT expr { Binop($1, Less, $3) } |
| expr LEQ expr { Binop($1, Leq, $3) } |
| expr GT expr { Binop($1, Greater, $3) } |
| expr GEQ expr { Binop($1, Geq, $3) } |
| ID ASSIGN expr { Assign($1, $3) } |
| ID WITH actuals_opt EXEC { Call($1, $3) } |
| ID IEXEC { Call($1, []) } |
| LPAREN expr RPAREN { $2 } |

actuals_opt:
/* nothing */ { [] } |
actuals_list { List.rev $1 } |

actuals_list:
expr { [$1] } |
actuals_list COMMA expr { $3 :: $1 } |

scanner.mll

{ open Parser } |
rule token = parse
[' ' 't' 'r' 'n'] { token lexbuf } (* Whitespace *) |
'/*' { comment lexbuf } (* Comments *) |
'(' { LPAREN } |
')' { RPAREN } |
'{' { LBRACE } |
'}' { RBRACE } |
';' { SEMI } |
',' { COMMA } |
'+' { PLUS } |
'- ' { MINUS } |
'* ' { TIMES } |
'/ ' { DIVIDE } |
'=' { ASSIGN } |
'==' { EQ } |
'!==' { NEQ } |
'< ' { LT } |
'<= ' { LEQ } |
'> ' { GT } |
'>>=' { GEQ } |
'if ' { IF } |
'else ' { ELSE } |
'for ' { FOR } |
'while ' { WHILE } |
'return ' { RETURN } |
'int ' { INT } |
'routine ' { ROUTINE } |
'-> ' { WITH } |
'| ' { EXEC }
sdl.ml (open source)

(* sdlcaml - Objective Caml interface for the SDL library
 * Copyright (C) 1999, Jean-Christophe FILLIATRE, (C) 2006 Elliott Oti
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Library General Public
 * License version 2, as published by the Free Software Foundation.
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU Library General Public License version 2 for more details
 * (enclosed in the file LGPL).
 *)

(* Modifications by Elliott Oti (May 2006)
 * Module Audio added
 * Module Window added
 * Functions added to Module Video and Event
 * Non-SDL functions added to Module Draw: scale, scale_to, read_tga, load_tga,
 * make_mipmaps, make_sfont and sfont_print
 *)

exception SDL_failure of string

let _ = Callback.register_exception "SDL_failure" (SDL_failure "")

type byte_array = (int, Bigarray.int8_unsigned_elt, Bigarray.c_layout)
Bigarray.Array1.t

(* Initialization. *)

type init_flag =
  | TIMER
  | AUDIO
  | VIDEO
  | CDROM
  | JOYSTICK
| NOPARACHUTE (* Don't catch fatal signals *) |
| EVENTTHREAD (* Not supported on all OS's *) |
| EVERYTHING |

external init : init_flag list -> unit = "sdlstub_init"

external quit : unit -> unit = "sdlstub_quit"

external get_error : unit -> string = "sdlstub_get_error"

(******************************* Video. *****************************)
module Video = struct

  type video_flag =
      | SWSURFACE (* Surface is in system memory *)
      | HWSURFACE (* Surface is in video memory *)
      | ANYFORMAT (* Allow any video pixel format *)
      | HWPALETTE (* Surface has exclusive palette *)
      | DOUBLEBUF (* Set up double-buffered video mode *)
      | FULLSCREEN (* Surface is a full screen display *)
      | HWACCEL (* Blit uses hardware acceleration *)
      | SRCOLORKEY (* Blit uses a source color key *)
      | RLEACCEL (* Colorkey blit is RLE accelerated *)
      | SRCALPHA (* Blit uses source alpha blending *)
      | SRCCLIPPING (* Blit uses source clipping *)
      | OPENGL (* Surface supports OpenGL *)
      | RESIZABLE (* Surface is resizable *)
      | NOFRAME (* Creates a window with no title frame and no border *)

  type surface

  external free_surface : surface -> unit
           = "sdlstub_free_surface"
  external surface_pixels : surface -> byte_array
          = "sdlstub_surface_pixels"
  external surface_width : surface -> int
            = "sdlstub_surface_width"
  external surface_height : surface -> int
             = "sdlstub_surface_height"
  external surface_flags : surface -> video_flag list
             = "sdlstub_surface_flags"
  external surface_bpp : surface -> int
             = "sdlstub_surface_bpp"
  external surface_rmask : surface -> int
             = "sdlstub_surface_rmask"
  external surface_gmask : surface -> int
             = "sdlstub_surface_gmask"
  external surface_bmask : surface -> int
             = "sdlstub_surface_bmask"
  external surface_amask : surface -> int
             = "sdlstub_surface_amask"
  external must_lock : surface -> bool
             = "sdlstub_must_lock"
external lock_surface : surface -> unit
  = "sdlstub_lock_surface"
external unlock_surface : surface -> unit
  = "sdlstub_unlock_surface"
external video_mode_ok : int -> int -> int -> video_flag list -> bool
  = "sdlstub_video_mode_ok"
external set_video_mode : int -> int -> int -> video_flag list -> surface
  = "sdlstub_set_video_mode"
external create_rgb_surface : video_flag list -> int -> int -> surface
  = "sdlstub_create_rgb_surface"
external load_bmp : string -> surface
  = "sdlstub_load_bmp"
external save_bmp : surface -> string -> unit
  = "sdlstub_save_bmp"
external set_color_key : surface -> video_flag list -> int32 -> unit
  = "sdlstub_set_color_key"
external set_alpha : surface -> video_flag list -> int -> unit
  = "sdlstub_set_alpha"
external set_clipping : surface -> int -> int -> int -> int -> unit
  = "sdlstub_set_clipping"
let disable_clipping s = set_clipping s 0 0 0 0
external display_format : surface -> surface
  = "sdlstub_display_format"
external get_rgb : surface -> int32 -> int * int * int
  = "sdlstub_get_rgb"
external get_rgba : surface -> int32 -> int * int * int * int
  = "sdlstub_get_rgba"
external map_rgb : surface -> int -> int -> int -> int32
  = "sdlstub_map_rgb"
external map_rgba : surface -> int -> int -> int -> int
  = "sdlstub_map_rgba"

type rect = {
    mutable rect_x : int;
    mutable rect_y : int;
    mutable rect_w : int;
    mutable rect_h : int
}

external fill_surface : surface -> int32 -> unit
  = "sdlstub_fill_surface"
external fill_rect : surface -> rect -> int32 -> unit
  = "sdlstub_fill_rect"
external update_surface : surface -> unit
  = "sdlstub_update_surface"
external update_rect : surface -> int -> int -> int -> int -> unit
  = "sdlstub_update_rect"
external update_rects : surface -> rect array -> unit
  = "sdlstub_update_rects"
external flip : surface -> unit
  = "sdlstub_flip"
external blit_surface : surface -> rect option ->
surface -> rect option -> unit
= "sdlstub_blit_surface"

type color = {
  red : int;
  green : int;
  blue : int
}

external set_colors : surface -> color array -> int -> int -> bool
= "sdlstub_set_colors"
external show_cursor : bool -> unit
= "sdlstub_show_cursor"
external warp_mouse : int -> int -> unit
= "sdlstub_warp_mouse"
external string_of_pixels : surface -> string = "sdlstub_string_of_pixels"

end
(**********************************************************************
*************** End Video. **********************************************)
(**********************************************************************
*************** Window management ****************************************
module Window = struct
  external set_caption : string -> string -> unit = "sdlstub_set_caption"
  external get_caption : unit -> string * string = "sdlstub_get_caption"
  external set_icon : Video.surface -> unit = "sdlstub_set_icon"
  external iconify_window : unit -> unit = "sdlstub_iconify_window"
  external toggle_fullscreen : Video.surface -> unit = "sdlstub_toggle_fullscreen"
  external set_grab_input : bool -> unit = "sdlstub_set_grab_input"
  external get_grab_input : unit -> bool = "sdlstub_get_grab_input"
end
(**********************************************************************
*************** End Window management *******************************
(**********************************************************************
*************** Open GL support ****************************************
module SDLGL = struct
  type gl_attr =
    RED_SIZE |
    GREEN_SIZE |
    BLUE_SIZE |
    ALPHA_SIZE |
    DOUBLEBUFFER |
    BUFFER_SIZE |
    DEPTH_SIZE |
    STENCIL_SIZE |
    ACCUM_RED_SIZE |
    ACCUM_GREEN_SIZE |
    ACCUM_BLUE_SIZE |
    ACCUM_ALPHA_SIZE
external swap_buffers : unit -> unit = "sdlstub_GL_swap_buffers"

external load_bmp : string -> Video.surface = "sdlstub_GL_load_bmp"

external set_attribute : gl_attr -> int -> unit = "sdlstub_set_attribute"

external get_attribute : gl_attr -> int = "sdlstub_get_attribute"

end

(****************************************************************************** End Open GL support
******************************************************************************)

(****************************************************************************** Events.
******************************************************************************)

module Event = struct

type que_dis_ena = QUERY | DISABLE | ENABLE

type off_on = OFF | ON

type pointer

type app_state = APPMOUSEFOCUS | APPINPUTFOCUS | APPACTIVE

external get_app_state : unit -> (app_state list) = "sdlstub_get_app_state"

(* SDLKey enum *)


38
(*) SDLMod enum *

type key_mod = KMOD_NONE | KMOD_LSHIFT | KMOD_RSHIFT | KMOD_LCTRL | KMOD_RCTRL
| KMOD_LALT | KMOD_RALT | KMOD_LMETA | KMOD_RMETA
| KMOD_NUM | KMOD_CAPS | KMOD_MODE | KMOD_RESERVED

external enable_unicode : bool = "sdlstub_enable_unicode"

let default_repeat_delay = 500 and default_repeat_interval = 30

external enable_key_repeat : int -> int -> unit = "sdlstub_enable_key_repeat"
external get_mod_state : unit -> (key_mod list) = "sdlstub_get_mod_state"
external set_mod_state : key_mod list -> unit = "sdlstub_set_mod_state"
external get_key_name : key -> string = "sdlstub_get_key_name"

type press_release = RELEASED | PRESSED

type lost_gained = LOST | GAINED

type active_event = {
  focus: lost_gained;
  state: app_state
}

type keyboard_event = {
  keystate: press_release;
  scancode: int;
  sym: key;
  modifiers: key_mod list;
  unicode : int
}

type mouse_button = LEFT | MIDDLE | RIGHT | WHEELUP | WHEELDOWN

type mouse_motion_event = {
  mousestate : press_release;
  mx : int;
  my : int;
  mxrel : int;
  myrel : int
}
type mouse_button_event = {
    mousebutton: mouse_button;
    buttonstate: press_release;
    bx : int;
    by : int
}

type joy_axis_event = {
    which_axis : int;
    axis : int;
    jvalue : int
}

type joy_ball_event = {
    which_ball : int;
    ball : int;
    jxrel : int;
    jyrel : int
}

type joy_hat_event = {
    which_hat : int;
    hat : int;
    hvalue : int
}

type joy_button_event = {
    which_button : int;
    joybutton : int;
    jstate : press_release
}

type resize_event = {
    w : int;
    h : int
}

type user_event = {code : int; data1 : pointer; data2 : pointer}

type sys_wm_event

type event =
    | NoEvent
    | Active of active_event
    | Key of keyboard_event
    | Motion of mouse_motion_event
    | Button of mouse_button_event
    | Jaxis of joy_axis_event
    | Jball of joy_ball_event
    | Jhat of joy_hat_event
    | Jbutton of joy_button_event
    | Resize of resize_event
(* Event functions *)
external pump_events : unit -> unit = "sdlstub_pump_events"
external poll_event : unit -> event = "sdlstub_poll_event"
external wait_event : unit -> event = "sdlstub_wait_event"
end

(***************************** End Events. *******************************)

(***************************** Timer. *****
module Timer = struct

  external get_ticks : unit -> int
     = "sdlstub_get_ticks"
  external delay : int -> unit
     = "sdlstub_delay"

end

(***************************** End Timer. *******************************)

(****************************** Audio ****************************)
(* low-level audio. *)
module Audio = struct

  type sample_type =
    | U8
    | S8
    | U16
    | S16
    | U16LSB
    | S16LSB
    | U16MSB
    | S16MSB

type audio_status =
    | STOPPED
    | PAUSED
    | PLAYING
    | UNKNOWN

type channel_type =
    | MONO
    | STEREO
type audio_spec = {
  frequency: int;
  format: sample_type;
  channels: channel_type;
  silence: int;
  samples: int;
  size: int;
}

let rec int_of_sampletype t =
  match t with
  | U8       -> 0x0008 (* Unsigned 8-bit samples *)
  | S8       -> 0x8008 (* Signed 8-bit samples *)
  | U16LSB   -> 0x0010 (* Unsigned 16-bit samples *)
  | S16LSB   -> 0x8010 (* Signed 16-bit samples *)
  | U16MSB   -> 0x1010 (* As above, but big-endian byte order *)
  | S16MSB   -> 0x9010 (* As above, but big-endian byte order *)
  | U16      -> int_of_sampletype U16LSB
  | S16      -> int_of_sampletype S16LSB

let sampletype_of_int i =
  match i with
  | 0x0008 -> U8
  | 0x8008 -> S8
  | 0x0010 -> U16LSB
  | 0x8010 -> S16LSB
  | 0x1010 -> U16MSB
  | 0x9010 -> S16MSB
  | _     -> raise (SDL_failure "Unknown sample format")

let int_of_channel t =
  match t with
  | MONO    -> 1
  | STEREO  -> 2

let channel_of_int i =
  match i with
  | 1  -> MONO
  | 2  -> STEREO
  | _  -> raise (SDL_failure "Unknown channel format")

let mix_maxvolume = 128

external proto_open_audio : int -> int -> int -> int -> int * int * int * int * int = "sdlstub_open_audio"

let open_audio a callback =
  Callback.register "ml_setaudiocallback" callback;
let (fr, fo, ch, si, sa, sz) = proto_open_audio a.frequency
(int_of_sampletype a.format) (int_of_channel a.channels) a.samples in
{frequency = fr; format = (sampletype_of_int fo); channels =
(channel_of_int ch); silence = si; samples = sa; size = sz}

external close_audio : unit -> unit
= "sdlstub_close_audio"

external lock_audio : unit -> unit
= "sdlstub_lock_audio"

external unlock_audio : unit -> unit
= "sdlstub_unlock_audio"

external pause_audio : bool -> unit
= "sdlstub_pause_audio"

external proto_get_audio_status : unit -> int
= "sdlstub_get_audio_status"

external proto_load_wav : string -> int * int * int * int * int * int *
byte_array
= "sdlstub_load_wav"

let get_audio_status () =
  let r = proto_get_audio_status () in
  if r = 0 then STOPPED else
  if r = 1 then PAUSED else
  if r = 2 then PLAYING else
  UNKNOWN

let load_wav file =
  let (fr, fo, ch, si, sa, sz, buf) = proto_load_wav file in
  {frequency = fr; format = (sampletype_of_int fo); channels =
  (channel_of_int ch); silence = si; samples = sa; size = sz}, buf

external free_wav : byte_array -> unit
= "sdlstub_free_wav"

external mix_audio : byte_array -> byte_array -> int -> unit
= "sdlstub_mix_audio"

external proto_convert_audio : int -> int -> int -> int -> int ->
byte_array -> byte_array
= "sdlstub_convert_audio_byte" "sdlstub_convert_audio"

let convert_audio f_fmt f_ch f_fr fmt ch fr ain =
  proto_convert_audio (int_of_sampletype f_fmt) (int_of_channel f_ch) f_fr
  (int_of_sampletype fmt) (int_of_channel ch) fr ain

external proto_fx_pan : float -> float -> byte_array -> byte_array -> unit
= "fxstub_pan"
let fx_pan pan volume sample =
  lock_audio ();
  let len = Bigarray.Array1.dim sample in
  let newsample = Bigarray.Array1.create Bigarray.int8_unsigned
  Bigarray.c_layout len in
  proto_fx_pan pan volume sample newsample;
  unlock_audio ();
  newsample

external proto_fx_shift : float -> byte_array -> byte_array -> int
  = "fxstub_shift"

let fx_shift pitch sample =
  lock_audio ();
  let p = if pitch < 0.1 then 0.1 else if pitch > 10.0 then 10.0 else pitch
  in
  let len = float_of_int (Bigarray.Array1.dim sample) in
  let newsample = Bigarray.Array1.create Bigarray.int8_unsigned
  Bigarray.c_layout (int_of_float (len /. p)) in
  let _ = proto_fx_shift p sample newsample in
  unlock_audio ();
  newsample
end

(****************************** End Audio ******************************)

(****************************** Extra SDL-related but non-SDL core routines ******************************)

(****************************** Draw ******************************)

module Draw = struct

    exception TGA_failure of string
    exception Sfont_failure of string

    type filter = BOX of int | TRIANGLE of int | BELL of int | BSPLINE of int | HERMITE of int | MITCHELL of int | LANCZOS3 of int

    let box = BOX 1
    let triangle = TRIANGLE 2
    let bell = BELL 3
    let bspline = BSPLINE 4
    let hermite = HERMITE 5
    let mitchell = MITCHELL 6
    let lanczos3 = LANCZOS3 7

    type sfont = {
        font_list: (int * Video.rect) list;
        font_surf: Video.surface;
        font_space: int;
        font_letters: int;
    }
external put_pixel : Video.surface -> int -> int -> int32 -> unit
  = "sdldraw_put_pixel"

external get_pixel : Video.surface -> int -> int -> int32
  = "sdldraw_get_pixel"

type tga_orientation = From_upper_left | From_lower_left

let input_int16 ic =
  let lo = input_byte ic in
  let hi = input_byte ic in
  (hi lsl 8) lor lo

(* Targa TGA image file reader, based on the specs at
http://astronomy.swin.edu.au/~pbourke/dataformats/tga/
    Takes as parameter the file name and returns a tuple containing the image
    width, height, bytes-per-pixel
    and a string containing the image data in BGR(A) format.
    Reads 15, 16, 24 and 32 bit-per-pixel raw and RLE-compressed images.
    Throws TGA_exception when anything goes wrong. *)
let read_tga file =
  try
    let ic = open_in_bin file in
    (* Read in TGA header *)
    let idlength = input_byte ic in
    let (* colourmaptype *) _ = input_byte ic in
    let datatypecode = input_byte ic in
    let (* colourmaporigin *) _ = input_int16 ic in
    let (* colourmaplength *) _ = input_int16 ic in
    let (* colourmapdepth  *) _ = input_byte ic in
    let (* x_origin *) _ = input_int16 ic in
    let (* y_origin *) _ = input_int16 ic in
    let width = input_int16 ic in
    let height = input_int16 ic in
    let bitsperpixel = input_byte ic in
    let imagedescriptor = input_byte ic in
    let rec consume_id cnt =
      if (cnt > 0) then let _ = input_byte ic in consume_id (cnt - 1)
    in
    consume_id idlength;
    (* Read in TGA data *)
    let bpp = bitsperpixel/8 in
    let len = width*height*bpp in
    let data = String.make len ' ' in
    let rec decode_run byte pos =
      let rtype = (byte land 0x80) lsr 7
      and rlen = ((byte land 0x7F) + 1) in
      let newpos = pos + rlen * bpp in
      if rtype = 0 then (* RAW unencoded pixels *)
        really_input ic data pos (rlen*bpp)
else (* Run length encoded pixels *)
  begin
    let b = input_byte ic
    and g = input_byte ic
    and r = if bpp > 2 then input_byte ic else -1
    and a = if bpp > 3 then input_byte ic else -1 in
    for i = 0 to rlen - 1 do
      String.set data (pos + bpp*i + 0) (char_of_int b);
      String.set data (pos + bpp*i + 1) (char_of_int g);
      if bpp > 2 then String.set data (pos + bpp*i + 2) (char_of_int r);
      if bpp > 3 then String.set data (pos + bpp*i + 3) (char_of_int a);
    done;
  end;
if newpos < String.length data then decode_run (input_byte ic) (newpos) else ();
in
if datatypecode = 2 then
  really_input ic data 0 len
else if datatypecode = 10 then
  decode_run (input_byte ic) 0
else raise (TGA_failure "Cannot decode this TGA file type");
close_in ic;
let orientation =
  if imagedescriptor land 0x20 = 0
    then From_lower_left
    else From_upper_left
in
(width, height, bitsperpixel, data, orientation);
with
  _ -> raise (TGA_failure "Unable to load TGA file")

(* Returns an RGBA tuple representing the pixel in string s at position (x,y),
  with texturemap width w and depth bitsperpixel
  FIXME: Only checked for 24 and 32 bpp; 15 and 16 bpp untested *)
let get_tga_pixel s x y w bitsperpixel =
  let bpp = bitsperpixel / 8 in
  let b = int_of_char (String.get s (0 + (y*w + x) * bpp))
  and g = int_of_char (String.get s (1 + (y*w + x) * bpp))
  and r = if bitsperpixel > 16 then int_of_char (String.get s (2 + (y*w + x) * bpp)) else 0
  and a = if bitsperpixel > 24 then int_of_char (String.get s (3 + (y*w + x) * bpp)) else 1
  in
  if bitsperpixel > 16 then
    (r, g, b, a)
  else if bitsperpixel = 15 then
    ((r lsl 1) land 0xF8), ((r lsl 5) + ((g land 0xE0) lsr 2)), (g lsl 3), (r land 0x80) lsr 7 )
  else

let load_tga file =
  let w, h, bitsperpixel, s, orientation = read_tga file in
  let surf = Video.create_rgb_surface [Video.SWSURFACE] w h bitsperpixel in
  for y = 0 to (h - 1) do
    let ysurf =
      begin
        match orientation with
        | From_upper_left -> y
        | From_lower_left -> (h - 1 - y)
      end
      in
    for x = 0 to (w - 1) do
      begin
        let (r, g, b, a) = get_tga_pixel s x y w bitsperpixel in
        put_pixel surf x ysurf (Video.map_rgba surf r g b a);
      end;
    done;
  done;
  surf

let make_sfont surf =
  let ascii_start = 33
  and pink = Video.map_rgba surf 255 0 255 255
  and w = Video.surface_width surf
  and h = (Video.surface_height surf) - 1 in
  let rec make_sfont_list lastpink ch x x1 x2 =
    if x >= w then []
    else
      begin
        let pixel = get_pixel surf x 0 in
        let ispink = (pixel = pink) in
        match lastpink, ispink with
        | true, true -> make_sfont_list ispink ch (x+1) x1 x2;
        | true, false -> make_sfont_list ispink ch (x+1) x x2;
        | false, true -> (ch, {Video.rect_x = x1; Video.rect_y = 1;
          Video.rect_w = (x - x1); Video.rect_h = h}):(make_sfont_list ispink (ch+1) (x+1) x x);
        | false, false -> make_sfont_list ispink ch (x+1) x x2;
      end;
    in
let l = make_sfont_list true ascii_start 0 0 0 in
let letter_L = List.assoc (int_of_char 'L') l
and letter_spc = List.assoc (int_of_char '!') l in
let fs = letter_L.Video.rect_w
and fl = letter_spc.Video.rect_w in
(font_list = l; font_surf = surf; font_space = fs; font_letters = fl;
font_line = h )

(* Prints string s at location [x,y] with font "font" on surface dest *)
let sfont_print s x y font dest =
let offx = ref x
and offy = ref y in
let spr c =
match c with
| ' ' -> offx := !offx + font.font_space;
| '
' -> offy := !offy + font.font_line; offx := x;
| _ -> begin
let r = List.assoc (int_of_char c)
font.font_list in
Video.blit_surface font.font_surf (Some r)
dest (Some {Video.rect_x = !offx; Video.rect_y = !offy; Video.rect_w =
(Video.surface_width font.font_surf); Video.rect_h = (Video.surface_height
font.font_surf)});

offx := !offx + r.Video.rect_w +
font.font_letters;
end;

in
String.iter spr s

(******************* Bitmap scaling **********************)
let print_array a =
let dim = Array.length a.(0) in
for i = 0 to (dim - 1) do
for j = 0 to (dim - 1) do
Printf.printf "%f
" a.(i).(j);
done;
Printf.printf "\n";
done

let normalize a =
let total = ref 0.0 in
let dim = Array.length a.(0) in
for i = 0 to (dim - 1) do
for j = 0 to (dim - 1) do

total := !total +. a.(i).(j);
done;
done;
for i = 0 to (dim - 1) do
for j = 0 to (dim - 1) do
a.(i).(j) <- (a.(i).(j) /. !total);
done;
let dist x y dim = 
  let x2 = (float_of_int x) -. ((float_of_int dim) /. 2.0) 
  and y2 = (float_of_int y) -. ((float_of_int dim) /. 2.0) 
  in (sqrt ((x2 *. x2) +. (y2 *. y2)))

let box_filter dim = 
  normalize (Array.make_matrix dim dim 1.0)

let tent_filter dim = 
  let t = dist dim dim dim in 
  let f x =  (t -. (abs_float x)) /. t in 
  let a = Array.make_matrix (dim+1) (dim+1) 0.0 in 
  for i = 0 to dim do 
    for j = 0 to dim do 
      a.(i).(j) <- f (dist i j dim); 
    done; 
  done; 
  normalize a

let lanczos3_filter dim = 
  let f x = 
    if (abs_float x) > 3.0 then 0.0 else 
    let pi = 4.0 *. atan 1.0 in 
    let pix' = pi *. x in 
    let pix = if (abs_float pix') > 0.1 then pix' else 0.1 in 
    let pix3 = pix /. 3.0 in 
    ((sin pix)/. pix) *. ((sin pix3) /. pix3) 
  in 
  let a = Array.make_matrix (dim+1) (dim+1) 0.0 in 
  let dim2 = dist dim dim dim in 
  for i = 0 to dim do 
    for j = 0 to dim do 
      a.(i).(j) <- f ((dist i j dim) *. 3.0 /. dim2); 
    done; 
  done; 
  normalize a

let create_filter filter dim = 
  if filter = box then (box_filter dim) 
  else if filter = triangle then (tent_filter dim) 
  else lanczos3_filter (dim + 4)

let round f = 
  (int_of_float (f +. 0.5))

let pixel_round f = 
  let c = round f in 
  if c > 255 then 255 else if c < 0 then 0 else c
let convolute kernel s x y =
let r = ref 0.0 and g = ref 0.0 and b = ref 0.0 and a = ref 0.0 in
let h = (Video.surface_height s)
and w = (Video.surface_width s) in
let len = (Array.length kernel.(0)) in
let halflen = len/2 in
for i = (x + halflen - len + 1) to (x + halflen) do
  for j = (y + halflen - len + 1) to (y + halflen) do
    let k = i - (x + halflen - len + 1) in
    let l = j - (y + halflen - len + 1) in
    let x' = if i >= 0 then (if i < w then i else (w - 1)) else 0 in
    let y' = if j >= 0 then (if j < h then j else (h - 1)) else 0 in
    let p = get_pixel s x' y' in
    let (r',g',b',a') = Video.get_rgba s p in
    r := !r +. ((float_of_int r') *. kernel.(k).(l) );
    g := !g +. ((float_of_int g') *. kernel.(k).(l) );
    b := !b +. ((float_of_int b') *. kernel.(k).(l) );
    a := !a +. ((float_of_int a') *. kernel.(k).(l) );
  done;
let scale_to s w h filter =
  let t = Video.create_rgb_surface [Video.SWSURFACE] w h
  (Video.surface_bpp s) in
  let w' = (Video.surface_width s) and h' = (Video.surface_height s) in
  let fw = (float_of_int w')/.(float_of_int w) and fh = (float_of_int h')/.(float_of_int h) in
  let dim = if fw > 1.0 then fw else 1.0/.fw in
  let filter' = create_filter filter (round dim) in
  for i = 0 to ((Video.surface_width t) - 1) do
    for j = 0 to ((Video.surface_height t) - 1) do
      let si = (float_of_int i) *. fw and sj = (float_of_int j) *. fh in
      let p = convolute filter' s (int_of_float si) (int_of_float sj) in
      put_pixel t i j p;
    done;
  done;
  t

let scale s f filter =
  let w' = (Video.surface_width s) and h' = (Video.surface_height s) in
  scale_to s (int_of_float ((float_of_int w') *. f)) (int_of_float ((float_of_int h') *. f)) filter

  (* Takes in an Sdl.surface and a filter (type Sdl.Draw.t_filter) as argument,
    and returns an array of rectangular
    bitmaps with the sides' dimension a power of two. The largest bitmap is at
    offset 0, the
smallest at n - 1. If the original bitmap is square and the length of the
tiles are a power of two
then it will be placed unchanged into offset 0, else the bitmap will be
resized to the closest
suitable size and that will be used as the base *)

let make_mipmaps s filter =
let p2 n =
  let rec p2' n i =
    if n <= 1 then i else (p2' (n/2) (i+1))
in p2' n 0
in
let rec int_exp m e =
  if e < 0 then 0 else if e = 0 then 1 else m * (int_exp m (e-1))
in
let closest_power_of_2 n =
  let i = (p2 n) in
  let min = (int_exp 2 i) and max = (int_exp 2 (i+1)) in
  if (n - min) < (max - n) then min else max
in
let w = (Video.surface_width s) and h = (Video.surface_height s) in
let dimx' = w
and dimy' = h in
let dimx = (closest_power_of_2 dimx')
and dimy = (closest_power_of_2 dimy') in
let dim = if (dimx > dimy) then dimx else dimy in
let s2 = if (w = dimx && h = dimy) then s else (scale_to s dimx dimy filter) in
let num = ((p2 dim) + 1) in
let mipmaps = Array.make num s2 in
for i = 1 to (num - 1) do
  let dimx'' = (dimx/(int_exp 2 i))
  and dimy'' = (dimy/(int_exp 2 i)) in
  mipmaps.(i) <- (scale_to s dimx'' dimy'' filter);
done;
mipmaps
end

(**************************** End Draw ************************** *)

sdl_sub.c (open source)

/*
 * sdlcaml - Objective Caml interface for the SDL library
 * Copyright (C) 1999, Jean-Christophe FILLIATRE, (C) 2006, 2007, 2008 Elliott Oti
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Library General Public
 * License version 2, as published by the Free Software Foundation.
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
*/
*/ MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. *
* See the GNU Library General Public License version 2 for more details
* (enclosed in the file LGPL).
*/

 * Module Audio added
 * Module Window added
 * Functions added to Module Video and Event
 * Non-SDL functions added to Module Draw: scale, scale_to, read_tga, load_tga,
 * make_mipmaps, make_sfont and sfont_print
 */

#include <stdio.h>
#include <string.h>
#include <math.h>
#include <SDL/SDL.h>

/* Ugly: SDL_AudioSpec has a field called "callback" which caml/callback.h redefines to
caml_callback.
So this function has to come before the caml #includes*/
void set_audiospec(SDL_AudioSpec *in, int freq, int format, int channels, int samples,
void (*callback)(void *userdata, Uint8 *stream, int len))
{
    in->freq = freq;
    in->format = format;
    in->channels = channels;
    in->samples = samples;
    in->callback = callback;
    in->userdata = NULL;
}

/* CAML - C interface */
#include <caml/mlvalues.h>
#include <caml/memory.h>
#include <caml/alloc.h>
#include <caml/fail.h>
#include <caml/callback.h>
#include <caml/bigarray.h>

/* Caml list manipulations */
#define NIL_tag 0
#define CONS_tag 1

value nil(void)
{
    CAMLparam0();
    CAMLreturn (Val_int(0));
}
value cons(value x, value l)
{
    CAMLparam2(x, l);
    CAMLlocal1(m);
    m = alloc(2, CONS_tag);
    Store_field(m, 0, x);
    Store_field(m, 1, l);
    CAMLreturn (m);
}

int is_nil(value l)
{
    CAMLparam1(l);
    CAMLreturn (Is_long(l));
}

int is_not_nil(value l)
{
    CAMLparam1(l);
    CAMLreturn (Is_block(l));
}

value hd(value l)
{
    CAMLparam1(l);
    CAMLreturn (Field(l, 0));
}

value tl(value l)
{
    CAMLparam1(l);
    CAMLreturn (Field(l, 1));
}

/* conversion between OCAMLSDL flags and C SDL flags */
#define TIMER_tag 0
#define AUDIO_tag 1
#define VIDEO_tag 2
#define CDROM_tag 3
#define JOYSTICK_tag 4
#define NOPARACHUTE_tag 5
#define EVENTTHREAD_tag 6
#define EVERYTHING_tag 7

int init_flag_val(value flag_list)
{
    CAMLparam1(flag_list);
    int flag = 0;
    value l = flag_list;
    while (is_not_nil(l))
    {
        switch (Int_val(hd(l)))
        {
{
  case TIMER_tag : flag |= SDL_INIT_TIMER ; break;
  case AUDIO_tag : flag |= SDL_INIT_AUDIO ; break;
  case VIDEO_tag : flag |= SDL_INIT_VIDEO ; break;
  case CDROM_tag : flag |= SDL_INIT_CDROM ; break;
  case JOYSTICK_tag : flag |= SDL_INIT_JOYSTICK ; break;
  case NOPARACHUTE_tag : flag |= SDL_INIT_NOPARACHUTE ; break;
  case EVENTTHREAD_tag : flag |= SDL_INIT_EVENTTHREAD ; break;
  case EVERYTHING_tag : flag |= SDL_INIT_EVERYTHING ; break;
}
  l = tl(l);
}
CAMLreturn (flag);
}

#define SWSURFACE_tag 0
#define HWSURFACE_tag 1
#define ANYFORMAT_tag 2
#define HWPALETTE_tag 3
#define DOUBLEBUF_tag 4
#define FULLSCREEN_tag 5
#define HWACCEL_tag 6
#define SRCOLORKEY_tag 7
#define RLEACCEL_tag 8
#define SRCALPHA_tag 9
#define SRCCOLORING_tag 10
#define OPENGL_tag 11
#define RESIZABLE_tag 12
#define NOFRAME_tag 13

int video_flag_val(value flag_list)
{
  CAMLparam1(flag_list);
  int flag = 0;
  value l = flag_list;
  while (is_not_nil(l))
  {
    switch(Int_val(hd(l)))
    {
    case SWSURFACE_tag   : flag |= SDL_SWSURFACE   ; break;
    case HWSURFACE_tag   : flag |= SDL_HWSURFACE   ; break;
    case ANYFORMAT_tag   : flag |= SDL_ANYFORMAT   ; break;
    case HWPALETTE_tag   : flag |= SDL_HWPALETTE   ; break;
    case DOUBLEBUF_tag   : flag |= SDL_DOUBLEBUF   ; break;
    case FULLSCREEN_tag  : flag |= SDL_FULLSCREEN  ; break;
    case HWACCEL_tag     : flag |= SDL_HWACCEL     ; break;
    case SRCCOLORKEY_tag : flag |= SDL_SRCCOLORKEY ; break;
    case RLEACCEL_tag    : flag |= SDL_RLEACCEL    ; break;
    case SRCALPHA_tag    : flag |= SDL_SRCALPHA    ; break;
    case OPENGL_tag      : flag |= SDL_OPENGL      ; break;
    case RESIZABLE_tag   : flag |= SDL_RESIZABLE   ; break;
    case NOFRAME_tag     : flag |= SDL_NOFRAME     ; break;
    }
value val_video_flag(int flags)
{
    CAMLparam0();
    value l = nil();
    if (flags & SDL_SWSURFACE)  l = cons(Val_int(SWSURFACE_tag),l);
    if (flags & SDL_HWSURFACE)  l = cons(Val_int(HWSURFACE_tag),l);
    if (flags & SDL_ANYFORMAT)   l = cons(Val_int(ANYFORMAT_tag),l);
    if (flags & SDL_HWPALETTE)   l = cons(Val_int(HWPALETTE_tag),l);
    if (flags & SDL_DOUBLEBUF)   l = cons(Val_int(DOUBLEBUF_tag),l);
    if (flags & SDL_FULLSCREEN)  l = cons(Val_int(FULLSCREEN_tag),l);
    if (flags & SDL_HWACCEL)     l = cons(Val_int(HWACCEL_tag),l);
    if (flags & SDL_SRCCOLORKEY) l = cons(Val_int(SRCCOLORKEY_tag),l);
    if (flags & SDL_RLEACCEL)    l = cons(Val_int(RLEACCEL_tag),l);
    if (flags & SDL_SRCALPHA)    l = cons(Val_int(SRCAALPHA_tag),l);
    if (flags & SDL_OPENGL)      l = cons(Val_int(OPENGL_tag),l);
    CAMLreturn (l);
}

/* raising SDL_failure exception */

void raise_failure() {
    raise_with_string(*caml_named_value("SDL_failure"), SDL_GetError());
}

value sdlstub_init(value vf) {
    CAMLparam1(vf);
    int flags = init_flag_val(vf);
    if (SDL_Init(flags) < 0) raise_failure();
    CAMLreturn (Val_unit);
}

value sdlstub_quit(value u) {
    CAMLparam1(u);
    SDL_Quit();
    CAMLreturn (Val_unit);
}

value sdlstub_get_error(value u){
    CAMLparam1(u);
    CAMLlocal1(result);
    char *s = SDL_GetError();
    Store_field (result, 0, copy_string(s));
    CAMLreturn (result);
}

value sdlstub_must_lock(value s) {
    CAMLparam1(s);
int b = SDL_MUSTLOCK(((SDL_Surface*) s));
CAMLreturn (Val_bool(b));
}

value sdlstub_video_mode_ok(value vw, value vh, value vbpp, value vf) {
  CAMLparam4(vw,vh,vbpp, vf);
  int w = Int_val(vw);
  int h = Int_val(vh);
  int bpp = Int_val(vbpp);
  int flags = video_flag_val(vf);

  CAMLreturn (Val_bool(SDL_VideoModeOK(w,h,bpp,flags)));
}

value sdlstub_set_video_mode(value vw, value vh, value vbpp, value vf) {
  CAMLparam4(vw, vh, vbpp, vf);
  CAMLlocal1(r);
  int w = Int_val(vw);
  int h = Int_val(vh);
  int bpp = Int_val(vbpp);
  int flags = video_flag_val(vf);
  SDL_Surface* s;

  if ((s = SDL_SetVideoMode(w,h,bpp,flags)) == NULL) raise_failure();
  r = (value)s;
  CAMLreturn (r);
}

value sdlstub_load_bmp(value vfile) {
  CAMLparam1(vfile);
  char * file = String_val(vfile);
  SDL_Surface* s;

  if ((s = SDL_LoadBMP(file)) == NULL) raise_failure();
  CAMLreturn ((value) s);
}

value sdlstub_save_bmp(value s, value vfile) {
  CAMLparam2(s, vfile);
  char * file = String_val(vfile);

  if (SDL_SaveBMP((SDL_Surface*) s, file) < 0) raise_failure();
  CAMLreturn (Val_unit);
}

/* Code by Jeff Molofee's openGL tutorial */
SDL_Surface * GLLoadBMP(char *filename) {
  Uint8 *rowhi, *rowlo;
  Uint8 *tmpbuf, tmpch;
  SDL_Surface *image;
  int i, j;
image = SDL_LoadBMP(filename);
if (image == NULL) {
    fprintf(stderr, "Unable to load %s: %s\n", filename, SDL_GetError());
    raise_failure();
}

/* GL surfaces are upsidedown and RGB, not BGR :-) */
tmpbuf = (Uint8 *)malloc(image->pitch);
if (tmpbuf == NULL) {
    fprintf(stderr, "Out of memory\n");
    raise_failure();
}
rowhi = (Uint8 *)image->pixels;
rowlo = rowhi + (image->h * image->pitch) - image->pitch;
for (i=0; i<image->h/2; ++i) {
    for (j=0; j<image->w; ++j) {
        tmpch = rowhi[j*3];
        rowhi[j*3] = rowhi[j*3+2];
        rowhi[j*3+2] = tmpch;
        tmpch = rowlo[j*3];
        rowlo[j*3] = rowlo[j*3+2];
        rowlo[j*3+2] = tmpch;
    }
    memcpy(tmpbuf, rowhi, image->pitch);
    memcpy(rowhi, rowlo, image->pitch);
    memcpy(rowlo, tmpbuf, image->pitch);
    rowhi += image->pitch;
    rowlo -= image->pitch;
}
free(tmpbuf);
return(image);

value sdlstub_GL_load_bmp(value f) {
    CAMLparam1(f);
    CAMLreturn ((value)GLLoadBMP(String_val(f)));
}

value sdlstub_string_of_pixels(value s) {
    CAMLparam1(s);
    CAMLLocal1(v);
    SDL_Surface * surf = (SDL_Surface *) s;
    int n = surf->w * surf->h * surf->format->BytesPerPixel;
    v = alloc_string(n);
    memcpy(String_val(v), surf->pixels, n);
    CAMLreturn(v);
}

value sdlstub_set_color_key(value s, value vf, value vk) {
    CAMLparam3(s,vf,vk);
    int flag = video_flag_val(vf);
    unsigned int key = Int32_val(vk);
if (SDL_SetColorKey((SDL_Surface*) s, flag, key) < 0) raise_failure();
CAMLreturn(V_unit);
}

value sdlstub_set_alpha(value s, value vf, value va) {
  CAMLparam3(s,vf,va);
  int flags = video_flag_val(vf);
  int alpha = Int_val(va);

  if (SDL_SetAlpha((SDL_Surface*) s, flags, alpha) < 0) raise_failure();
  CAMLreturn(V_unit);
}

value sdlstub_set_clipping(value s, value vtop, value vleft, value vbottom, value vright) {
  CAMLparam5(s,vtop,vleft, vbottom, vright);
  SDL_Rect r;
  r.x = Int_val(vleft);
  r.y = Int_val(vtop);
  r.w = Int_val(abs(vright - vleft));
  r.h = Int_val(abs(vbottom - vtop));
  SDL_SetClipRect((SDL_Surface*) s, &r);
  CAMLreturn(V_unit);
}

value sdlstub_display_format(value s) {
  CAMLparam1(s);
  SDL_Surface* n;
  if ((n = SDL_DisplayFormat((SDL_Surface*) s)) == NULL)raise_failure();
  CAMLreturn((value)n);
}

value sdlstub_create_rgb_surface(value vflags, value vw, value vh, value vdepth) {
  CAMLparam4(vflags, vw, vh, vdepth);
  CAMLlocal1(r);
  SDL_Surface* s;
  Uint32 flags = Int_val(vflags);
  int width = Int_val(vw);
  int height = Int_val(vh);
  int depth = Int_val(vdepth);
  Uint32 rmask=0, gmask=0, bmask=0, amask=0;
  if(depth == 32){
    #if SDL_BYTEORDER == SDL_BIG_ENDIAN
      rmask = 0xff000000;
      gmask = 0x00ff0000;
      bmask = 0x0000ff00;
      amask = 0x000000ff;
    #else
      rmask = 0x000000ff;
      gmask = 0x0000ff00;
      bmask = 0x00ff0000;
      amask = 0xff000000;
    #endif
  }
#endif
} else {
    rmask = 0x0000ff;
    gmask = 0x00ff00;
    bmask = 0xff0000;
    amask = 0;
#endif

if(depth == 16) {
    #if SDL_BYTEORDER == SDL_BIG_ENDIAN
      rmask = 0xF800;
      gmask = 0x7E0;
      bmask = 0x1F;
      amask = 1;
    #else
      rmask = 0x1F;
      gmask = 0x7E0;
      bmask = 0xF800;
      amask = 0;
    #endif
  }
  if(depth == 15) {
    #if SDL_BYTEORDER == SDL_BIG_ENDIAN
      rmask = 0xFC00;
      gmask = 0x3E0;
      bmask = 0x1F;
      amask = 1;
    #else
      rmask = 0x1F;
      gmask = 0x3E0;
      bmask = 0xFC00;
      amask = 0x8000;
    #endif
  }
  s = SDL_CreateRGBSurface(flags, width, height, depth,
                           rmask, gmask, bmask, amask);
  if (s == NULL) raise_failure();
  r = (value)s;
  CAMLreturn(r);
}

value sdlstub_free_surface(value s) {
  CAMLparam1(s);
  SDL_FreeSurface((SDL_Surface*) s);
CAMLreturn (Val_unit);
}

value sdlstub_lock_surface(value s) {
    CAMLparam1(s);
    if (SDL_LockSurface((SDL_Surface*) s) < 0) raise_failure();
    CAMLreturn (Val_unit);
}

value sdlstub_unlock_surface(value s) {
    CAMLparam1(s);
    SDL_UnlockSurface((SDL_Surface*) s);
    CAMLreturn (Val_unit);
}

value sdlstub_surface_pixels(value ps) {
    CAMLparam1(ps);
    SDL_Surface *s = ((SDL_Surface*) ps);
    CAMLreturn (alloc_bigarray_dims(BIGARRAY_UINT8 | BIGARRAY_C_LAYOUT, 1, s->pixels,
    s->w * s->h * (s->format->BitsPerPixel/8)));
}

value sdlstub_surface_width(value s) {
    CAMLparam1(s);
    CAMLreturn (Val_int(((SDL_Surface*) s)->w));
}

value sdlstub_surface_height(value s) {
    CAMLparam1(s);
    CAMLreturn (Val_int(((SDL_Surface*) s)->h));
}

value sdlstub_surface_flags(value s) {
    CAMLparam1(s);
    CAMLreturn (val_video_flag(((SDL_Surface*) s)->flags));
}

value sdlstub_surface_bpp(value s) {
    CAMLparam1(s);
    CAMLreturn (Val_int(((SDL_Surface*) s)->format->BitsPerPixel));
}

value sdlstub_surface_rmask(value s) {
    CAMLparam1(s);
    CAMLreturn (Val_int(((SDL_Surface*) s)->format->Rmask));
}

value sdlstub_surface_gmask(value s) {
    CAMLparam1(s);
    CAMLreturn (Val_int(((SDL_Surface*) s)->format->Gmask));
}

value sdlstub_surface_bmask(value s) {
CAMLparam1(s);
CAMLreturn (Val_int(((SDL_Surface*) s)->format->Bmask));
}

value sdlstub_surface_amask(value s) {
CAMLparam1(s);
CAMLreturn (Val_int(((SDL_Surface*) s)->format->Amask));
}

value sdlstub_map_rgb(value s, value vr, value vg, value vb) {
    CAMLparam4(s, vr, vg, vb);
    CAMLlocal1(rs);
    int r = Int_val(vr);
    int g = Int_val(vg);
    int b = Int_val(vb);
    int pixel = SDL_MapRGB(((SDL_Surface*) s)->format, r, g, b);
    rs = caml_copy_int32(pixel);
    CAMLreturn(rs);
}

value sdlstub_map_rgba(value s, value vr, value vg, value vb, value va) {
    CAMLparam5(s, vr, vg, vb, va);
    CAMLlocal1(rs);
    int r = Int_val(vr);
    int g = Int_val(vg);
    int b = Int_val(vb);
    int a = Int_val(va);
    int pixel = 0;
    pixel = SDL_MapRGBA(((SDL_Surface*) s)->format, r, g, b, a);
    rs = caml_copy_int32(pixel);
    CAMLreturn(rs);
}

value sdlstub_get_rgb(value s, value pixel){
    CAMLparam2(s,pixel);
    CAMLlocal1(result);
    Uint8 r =0, g=0, b=0;
    SDL_GetRGB(Int32_val(pixel), ((SDL_Surface *)s)->format, &r, &g, &b);
    result = caml_alloc(3,0);
    Store_field(result, 0, Val_int(r));
    Store_field(result, 1, Val_int(g));
    Store_field(result, 2, Val_int(b));
    CAMLreturn(result);
}

value sdlstub_get_rgba(value s, value pixel){
    CAMLparam2(s,pixel);
    CAMLlocal1(result);
    Uint8 r =0, g=0, b=0, a=0;
    SDL_GetRGBA(Int32_val(pixel), ((SDL_Surface *)s)->format, &r, &g, &b, &a);
    result = caml_alloc(4,0);
    Store_field(result, 0, Val_int(r));
    Store_field(result, 1, Val_int(g));
    Store_field(result, 2, Val_int(b));
    Store_field(result, 3, Val_int(a));
    CAMLreturn(result);
}
Store_field(result, 2, Val_int(b));
Store_field(result, 3, Val_int(a));
CAMLreturn(result);
}

value sdlstub_get_ticks(value u) {
  CAMLparam1(u);
  CAMLreturn (Val_int(SDL_GetTicks()));
}

value sdlstub_delay(value vms) {
  CAMLparam1(vms);
  int ms = Int_val(vms);
  SDL_Delay(ms);
  CAMLreturn (Val_unit);
}

value sdlstub_fill_surface(value s, value vc) {
  CAMLparam2(s,vc);
  int c = Int32_val(vc);
  if (SDL_FillRect((SDL_Surface*) s, NULL, c) < 0) raise_failure();
  CAMLreturn(Val_unit);
}

value sdlstub_fill_rect(value s, value vr, value vc) {
  CAMLparam3(s, vr, vc);
  int c = Int32_val(vc);
  SDL_Rect r;
  r.x = Int_val(Field(vr,0));
  r.y = Int_val(Field(vr,1));
  r.w = Int_val(Field(vr,2));
  r.h = Int_val(Field(vr,3));
  if (SDL_FillRect((SDL_Surface*) s, &r, c) < 0) raise_failure();
  CAMLreturn(Val_unit);
}

value sdlstub_update_surface(value s) {
  CAMLparam1(s);
  SDL_UpdateRect((SDL_Surface*) s, 0, 0, 0, 0);
  CAMLreturn (Val_unit);
}

value sdlstub_update_rect(value s, value vx, value vy, value vw, value vh) {
  CAMLparam5(s, vx, vy, vw, vh);
  int x = Int_val(vx);
  int y = Int_val(vy);
  int w = Int_val(vw);
  int h = Int_val(vh);
  SDL_UpdateRect((SDL_Surface*) s, x, y, w, h);
  CAMLreturn (Val_unit);
value sdlstub_update_rects(value s, value vn, value arr) {
    CAMLparam3(s, vn, arr);
    int n = Int_val(vn);
    value v;
    int i;
    SDL_Rect* rects;

    if (Wosize_val(arr) < n) invalid_argument("update_rects");
    rects = (SDL_Rect*) malloc(n * sizeof(SDL_Rect));
    for (i = 0; i < n; i++) {
        v = Field(arr, i);
        rects[i].x = Int_val(Field(v, 0));
        rects[i].y = Int_val(Field(v, 1));
        rects[i].w = Int_val(Field(v, 2));
        rects[i].h = Int_val(Field(v, 3));
    }
    SDL_UpdateRects((SDL_Surface*) s, n, rects);
    free(rects);
    CAMLreturn (Val_unit);
}

value sdlstub_flip(value s) {
    CAMLparam1(s);
    if (SDL_Flip((SDL_Surface*) s) < 0) raise_failure();
    CAMLreturn(Val_unit);
}

SDL_Rect* rect_from_option(value v, SDL_Rect* r) {
    CAMLparam1(v);
    value vr;
    if (v == Val_int(0)) {
        /* None */
        return (NULL);
    } else {
        /* Some */
        vr = Field(v, 0);
        r->x = Int_val(Field(vr, 0));
        r->y = Int_val(Field(vr, 1));
        r->w = Int_val(Field(vr, 2));
        r->h = Int_val(Field(vr, 3));
        return (r);
    }
}

/* assumption: v = Some rect and r is not NULL */
void update_rect_option(value v, SDL_Rect* r) {
    CAMLparam1(v);
    value vr = Field(v, 0);
    modify(&Field(vr, 0), Val_int(r->x));
    modify(&Field(vr, 1), Val_int(r->y));
    modify(&Field(vr, 2), Val_int(r->w));
    modify(&Field(vr, 3), Val_int(r->h));
}
value sdlstub_blit_surface(value src, value srcr, value dst, value dstr) {
  CAMLparam4(src, srcr, dst, dstr);
  SDL_Rect sr,dr;
  SDL_Rect *srp, *drp;

  srp = rect_from_option(srcr,&sr);
  drp = rect_from_option(dstr,&dr);

  if (SDL_BlitSurface((SDL_Surface*) src, srp, (SDL_Surface*) dst, drp) < 0)
    raise_failure();
  if (! (srp == NULL)) update_rect_option(srcr,srp);
  if (! (drp == NULL)) update_rect_option(dstr,drp);
  CAMLreturn(Val_unit);
}

value sdlstub_set_colors(value s, value arr, value vfirst, value vn) {
  CAMLparam4(s, arr, vfirst, vn);
  int ncolors = Int_val(vn);
  int firstcolor = Int_val(vfirst);
  int result, i;
  value v;
  SDL_Color* colors;

  if (ncolors < 0 || ncolors > Wosize_val(arr)) invalid_argument("set_colors");
  colors = (SDL_Color*) malloc(ncolors * sizeof(SDL_Color));
  if (colors == NULL) raise_out_of_memory();
  for (i=0; i<ncolors; i++) {
    v = Field(arr,i);
    colors[i].r = Int_val(Field(v,0));
    colors[i].g = Int_val(Field(v,1));
    colors[i].b = Int_val(Field(v,2));
  }
  result = SDL_SetColors((SDL_Surface*) s, colors, firstcolor, ncolors);
  free(colors);
  CAMLreturn (Val_bool(result));
}

value sdlstub_show_cursor(value vtoggle) {
  CAMLparam1(vtoggle);
  int toggle = Bool_val(vtoggle);

  SDL_ShowCursor(toggle);
  CAMLreturn(Val_unit);
}

value sdlstub_warp_mouse(value x, value y) {
  CAMLparam2(x,y);
  int lx = Int_val(x);
  int ly = Int_val(y);
  SDL_WarpMouse(x,y);
}
CAMLreturn(Val_unit);

/* --------------------- events -------------------------------- */
#define FLAG_TO_MOD_SIZE 12

int ML_flags_to_mask(value flags, int flag_to_cvalue[])
{
    int i, n;
    int mask = 0;
    CAMLparam0();
    while(Is_block(flags))
    {
        mask |= flag_to_cvalue[Int_val(Field(flags, 0))];
        flags = Field(flags, 1);
    }
    CAMLreturn(mask);
}

value carray_to_ML_list(int carray[], int arr_size)
{
    int i;
    CAMLparam0();
    CAMLlocal2(toreturn, tail);
    if(arr_size == 0)
        CAMLreturn(Val_int(0));
    toreturn = alloc(2, 0);
    Store_field(toreturn, 0, Val_int(carray[0]));
    tail = toreturn;
    for(i=1; i<arr_size; i++)
    {
        Field(tail, 1) = alloc(2, 0);
        tail = Field(tail, 1);
        Store_field(tail, 0, Val_int(carray[i]));
    }
    Store_field(tail, 1, Val_int(0));
    CAMLreturn(toreturn);
}

value mask_to_ML_flags(int mask, int flag_to_cvalue[], int arr_size)
{
    #ifdef _WIN32
        int flagar[1024];
    #else
        int flagar[arr_size];
    #endif
    int n=0, i;
    CAMLparam0();
    for(i=0; i<arr_size; i++)
    {
        if((mask & flag_to_cvalue[i]))
            flagar[n++] = i;
    }
CAMLreturn(carray_to_ML_list(flagar, n));
}

int key_to_flag[] =
{
    0, 0, 0, 0, 0, 0, 0, 0,
    2, 3, 0, 0,
    4, 5, 0, 0, 0, 0, 0,
    6, 0, 0, 0, 0, 0, 0,
    7, 0, 0, 0, 0,
    8, 9, 10, 11, 12, 0,
    13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71,
    0, 0, 0, 0,
    72,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 0, 0, 0,
    210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232,
    232
};

int flag_to_key[] =
{
};
int flag_to_mod[] = {
    0x0000, 0x0001, 0x0002, 0x0040, 0x0080, 0x0100, 0x0200, 0x0400, 0x0800, 0x1000, 0x2000, 0x4000, 0x8000
};

int flag_to_appstate[] = {
    0x01, 0x02, 0x04
};

#define FLAG_TO_APPSTATE_SIZE 3

value sdlstub_get_app_state(value u)
{
    CAMLparam1(u);
    CAMLreturn(mask_to_ML_flags(SDL_GetAppState(),
                                flag_to_appstate, FLAG_TO_APPSTATE_SIZE));
}

value SDL_event_to_ML_tevent(SDL_Event event)
{
    CAMLparam0();
    CAMLlocal2(ML_event, to_return);
    int i;

    switch (event.type)
    {
    case SDL_ACTIVEEVENT:
    {
        ML_event=alloc(2, 0);
        Store_field(ML_event, 0, Val_int(event.active.gain));
        if (event.active.state & SDL_APPACTIVE)
            Store_field(ML_event, 1, Val_int(2));
        else if (event.active.state & SDL_APPINPUTFOCUS)
            Store_field(ML_event, 1, Val_int(1));
        else
            Store_field(ML_event, 1, Val_int(0));
        to_return=alloc(1, 0);
        Store_field(to_return, 0, ML_event);
        CAMLreturn(to_return);
    }
    case SDL_KEYDOWN:
    case SDL_KEYUP:
    {
        ML_event=alloc(5, 0);
        Store_field(ML_event, 0, Val_int(event.key.state));
        Store_field(ML_event, 1, Val_int(event.key.keysym.scancode));
        Store_field(ML_event, 2, Val_int(key_to_flag[event.key.keysym.sym]));
        Store_field(ML_event, 3,
                    mask_to_ML_flags(event.key.keysym.mod, flag_to_mod, 12));
        Store_field(ML_event, 4, Val_int(event.key.keysym.unicode));
        to_return=alloc(1, 1);
        Store_field(to_return, 0, ML_event);
    }
    }
CAMLreturn(to_return);
}

case SDL_MOUSEMOTION:
{
    ML_event=alloc(5,0);
    switch(event.motion.state)
    {
        case 0: Store_field(ML_event, 0, Val_int(0));break;
        case 1: Store_field(ML_event, 0, Val_int(1));break;
        default: Store_field(ML_event, 0, Val_int(2));break;
    }
    Store_field(ML_event, 1, Val_int(event.motion.x));
    Store_field(ML_event, 2, Val_int(event.motion.y));
    Store_field(ML_event, 3, Val_int(event.motion.xrel));
    Store_field(ML_event, 4, Val_int(event.motion.yrel));

    to_return=alloc(1,2);
    Store_field(to_return, 0, ML_event);
    CAMLreturn(to_return);
}

case SDL_MOUSEBUTTONDOWN:
case SDL_MOUSEBUTTONUP:
{
    ML_event=alloc(4,0);
    Store_field(ML_event, 0, Val_int(event.button.button - 1));
    Store_field(ML_event, 1, Val_int(event.button.state));
    Store_field(ML_event, 2, Val_int(event.button.x));
    Store_field(ML_event, 3, Val_int(event.button.y));

    to_return=alloc(1,3);
    Store_field(to_return, 0, ML_event);
    CAMLreturn(to_return);
}

case SDL_JOYAXISMOTION:
{
    ML_event=alloc(3,0);
    Store_field(ML_event, 0, Val_int(event.jaxis.which));
    Store_field(ML_event, 1, Val_int(event.jaxis.axis));
    Store_field(ML_event, 2, Val_int(event.jaxis.value));
    to_return=alloc(1,4);
    Store_field(to_return, 0, ML_event);
    CAMLreturn(to_return);
}

case SDL_JOYBALLMOTION:
{
    ML_event=alloc(4,0);
    Store_field(ML_event, 0, Val_int(event.jball.which));
    Store_field(ML_event, 1, Val_int(event.jball.ball));
    Store_field(ML_event, 2, Val_int(event.jball.xrel));
    Store_field(ML_event, 3, Val_int(event.jball.yrel));
    to_return=alloc(1,5);
    Store_field(to_return, 0, ML_event);
CAMLreturn(to_return);
}
case SDL_JOYHATMOTION:
{
    ML_event=alloc(3,0);
    Store_field(ML_event, 0, Val_int(event.jhat.which));
    Store_field(ML_event, 1, Val_int(event.jhat.hat));
    Store_field(ML_event, 2, Val_int(event.jhat.value));
    to_return=alloc(1,6);
    Store_field(to_return, 0, ML_event);
    CAMLreturn(to_return);
}
case SDL_JOYBUTTONUP:
case SDL_JOYBUTTONDOWN:
{
    ML_event=alloc(3,0);
    Store_field(ML_event, 0, Val_int(event.jbutton.which));
    Store_field(ML_event, 1, Val_int(event.jbutton.button));
    Store_field(ML_event, 2, Val_int(event.jbutton.state));
    to_return=alloc(1,7);
    Store_field(to_return, 0, ML_event);
    CAMLreturn(to_return);
}
case SDL_VIDEORESIZE:
{
    ML_event=alloc(2,0);
    Store_field(ML_event, 0, Val_int(event.resize.w));
    Store_field(ML_event, 1, Val_int(event.resize.h));
    to_return=alloc(1,8);
    Store_field(to_return, 0, ML_event);
    CAMLreturn(to_return);
}
case SDL_VIDEOEXPOSE:
{
    CAMLreturn(Val_int(1));
}
case SDL_QUIT:
{
    CAMLreturn(Val_int(2));
}
case SDL_USEREVENT:
/* ... */
case SDL_NUMEVENTS-1:
{
    ML_event=alloc(3,0);
    Store_field(ML_event, 0, Val_int(event.user.code));
    Store_field(ML_event, 1, (value)(event.user.data1));
    Store_field(ML_event, 2, (value)(event.user.data2));
    to_return=alloc(1,9);
    Store_field(to_return, 0, ML_event);
    CAMLreturn(to_return);
}
 case SDL_SYSWMEVENT:
  {
    to_return=alloc(1,10);
    Store_field(to_return, 0, (value)(event.syswm.msg));
    CAMLreturn(to_return);
  }
default:
  {
    fprintf(stderr,"Unknown event.\n");
    exit(-1);
  }
}

value sdlstub_poll_event(value u)
{
    SDL_Event event;
    CAMLparam1(u);
    int isevent;
    isevent=SDL_PollEvent(&event);
    if (isevent==1)
        CAMLreturn (SDL_event_to_ML_tevent(event));
    else
        CAMLreturn (Val_int(0));
}

value sdlstub_wait_event(value u)
{
    CAMLparam1(u);
    SDL_Event event;
    int isevent;
    isevent=SDL_WaitEvent(&event);
    if (isevent==1)
        CAMLreturn (SDL_event_to_ML_tevent(event));
    else
        CAMLreturn (Val_int(0));
}

void sdlstub_pump_events(value u)
{
    CAMLparam1(u);
    SDL_PumpEvents();
    CAMLreturn0;
}

value sdlstub_event_state(value type, value state)
{
    CAMLparam2(type, state);
    CAMLreturn(Val_int(1+SDL_EventState(Int_val(type), Int_val(state)-1)));
}

value sdlstub_get_mouse_state(value u)
{
    CAMLparam1(u);
CAMLlocall(toreturn);
int x,y,but;
toreturn=alloc(3,0);
but=SDL_GetMouseState(&x,&y);
switch(but)
{
    case 0: Store_field(toreturn, 0, Val_int(0));break;
    case 1: Store_field(toreturn, 0, Val_int(1));break;
    default: Store_field(toreturn, 0, Val_int(2));break;
}
Store_field(toreturn, 1, Val_int(x));
Store_field(toreturn, 2, Val_int(y));
CAMLreturn(toreturn);
}

value sdlstub_enable_unicode(value enable)
{
    CAMLparam1(enable);
    CAMLreturn(Val_int(SDL_EnableUNICODE(Int_val(enable)-1)+1));
}

void sdlstub_enable_key_repeat(value delay, value interval)
{
    CAMLparam2(delay, interval);
    if (SDL_EnableKeyRepeat(Int_val(delay), Int_val(interval)) == -1)
        raise_failure();
    else CAMLreturn0;
}

value sdlstub_get_mod_state(value u)
{
    CAMLparam1(u);
    CAMLreturn(mask_to_ML_flags(SDL_GetModState(),
flag_to_mod, FLAG_TO_MOD_SIZE));
}

void sdlstub_set_mod_state(value flags)
{
    CAMLparam1(flags);
    SDL_SetModState(ML_flags_to_mask(flags, flag_to_mod));
    CAMLreturn0;
}

value sdlstub_get_key_name(value ML_key)
{
    CAMLparam1(ML_key); /* perhaps GC likes CAMLlocall(toreturn); toreturn=* /
    CAMLreturn(copy_string(SDL_GetKeyName(flag_to_key[Int_val(ML_key)])));
}
value sdlstub_set_caption(value title, value icon)
{
    CAMLparam2(title, icon);
    SDL_WM_SetCaption(String_val(title), String_val(icon));
    CAMLreturn (Val_unit);
}

value sdlstub_get_caption(value u)
{
    CAMLparam1(u);
    CAMLlocal1(result);
    char *title, *icon;
    SDL_WM_GetCaption(&title, &icon);
    Store_field(result, 0, copy_string(title));
    Store_field(result, 1, copy_string(icon));
    CAMLreturn(result);
}

value sdlstub_set_icon(value s)
{
    CAMLparam1(s);
    SDL_WM_SetIcon((SDL_Surface *)s, NULL);
    CAMLreturn(Val_unit);
}

value sdlstub_iconify_window(value u)
{
    CAMLparam1(u);
    SDL_WM_IconifyWindow();
    CAMLreturn(Val_unit);
}

value sdlstub_toggle_fullscreen(value s)
{
    CAMLparam1(s);
    SDL_WM_ToggleFullScreen((SDL_Surface *)s);
    CAMLreturn(Val_unit);
}

value sdlstub_set_grab_input(value b)
{
    CAMLparam1(b);
    int vb = Int_val(b);
    SDL_GrabMode mode = vb? SDL_GRAB_ON : SDL_GRAB_OFF;
    SDL_WM_GrabInput(mode);
    CAMLreturn(Val_unit);
}

value sdlstub_get_grab_input(value u)
{
    CAMLparam1(u);
    CAMLlocal1(result);
```c
int r;
SDL_GrabMode mode = SDL_GRAB_QUERY;
r = SDL_WM_GrabInput(mode);
result = Bool_val(0);
if(r == SDL_GRAB_ON)result = Bool_val(1);
CAMLreturn(result);

/* open GL */
value sdlstub_GL_swap_buffers(value u) {
    CAMLparam1(u);
    SDL_GL_SwapBuffers();
    CAMLreturn(Val_unit);
}

SDL_GLattr  SDL_GLAttrArray[] = {
    SDL_GL_RED_SIZE ,
    SDL_GL_GREEN_SIZE ,
    SDL_GL_BLUE_SIZE ,
    SDL_GL_ALPHA_SIZE ,
    SDL_GL_DOUBLEBUFFER ,
    SDL_GL_BUFFER_SIZE ,
    SDL_GL_DEPTH_SIZE ,
    SDL_GL_STENCIL_SIZE ,
    SDL_GL_ACCUM_RED_SIZE ,
    SDL_GL_ACCUM_GREEN_SIZE ,
    SDL_GL_ACCUM_BLUE_SIZE ,
    SDL_GL_ACCUM_ALPHA_SIZE
};

value sdlstub_set_attribute(value a, value v) {
    CAMLparam2(a,v);
    int attr = Int_val(a);
    int val = Int_val(v);
    if(attr < sizeof(SDL_GLAttrArray)){
        SDL_GLattr  sdlattr = SDL_GLAttrArray[attr];
        SDL_GL_SetAttribute(sdlattr, val);
    }
    CAMLreturn(Val_unit);
}

value sdlstub_get_attribute(value a) {
    CAMLparam1(a);
    int attr = Int_val(a);
    int val = 0;
    if(attr < sizeof(SDL_GLAttrArray)){
        SDL_GLattr  sdlattr = SDL_GLAttrArray[attr];
        SDL_GL_GetAttribute(sdlattr, &val);
    }
}
```
CAMLreturn(Val_int(val));
}

/* audio */
static void __audio_callback(void *userdata, unsigned char *stream, int len)
{
    caml_callback(*caml_named_value("ml_setaudiocallback"),
    alloc_bigarray_dims(BIGARRAY_UINT8 | BIGARRAY_C_LAYOUT, 1, stream, len));
}

value sdlstub_open_audio(value freq, value format, value channels, value samples) {
    CAMLparam4 (freq, format, channels, samples);
    CAMLlocal1 (result);
    SDL_AudioSpec input, output;
    result = caml_alloc (6, 0);
    set_audiospec(&input, Int_val(freq), Int_val(format), Int_val(channels), Int_val(samples), __audio_callback);
    set_audiospec(&output, 0, 0, 0, 0, NULL);
    SDL_OpenAudio(&input, &output);
    Store_field(result, 0, Val_int((int)output.freq));
    Store_field(result, 1, Val_int((int)output.format));
    Store_field(result, 2, Val_int((int)output.channels));
    Store_field(result, 3, Val_int((int)output.silence));
    Store_field(result, 4, Val_int((int)output.samples));
    Store_field(result, 5, Val_int((int)output.size));
    CAMLreturn(result);
}

value sdlstub_close_audio(value u) {
    CAMLparam1(u);
    SDL_CloseAudio();
    CAMLreturn (Val_unit);
}

value sdlstub_load_wav(value file) {
    CAMLparam1(file);
    SDL_AudioSpec wav_spec;
    Uint32 wav_length;
    Uint8 *wav_buffer;
    char *filename = String_val(file);
    result = caml_alloc (7, 0);
    if( SDL_LoadWAV(filename, &wav_spec, &wav_buffer, &wav_length) == NULL ){
        caml_failwith(SDL_GetError());
    }
    Store_field(result, 0, Val_int((int)wav_spec.freq));
    Store_field(result, 1, Val_int((int)wav_spec.format));
    Store_field(result, 2, Val_int((int)wav_spec.channels));
    Store_field(result, 3, Val_int((int)wav_spec.silence));
    Store_field(result, 4, Val_int((int)wav_spec.samples));
Store_field (result, 5, Val_int((int)wav_spec.size));
Store_field (result, 6, alloc_bigarray_dims(BIGARRAY_UINT8 | BIGARRAY_C_LAYOUT, 1, wav_buffer, wav_length));
CAMLreturn (result);
}

value sdlstub_free_wav(value wav) {
  CAMLparam1(wav);
  SDL_FreeWAV(Data_bigarray_val(wav));
  CAMLreturn (Val_unit);
}

value sdlstub_pause_audio(value on) {
  CAMLparam1(on);
  int p = 0;
  if (Bool_val(on)) p = 1;
  SDL_PauseAudio(p);
  CAMLreturn (Val_unit);
}

value sdlstub_lock_audio(value u) {
  CAMLparam1(u);
  SDL_LockAudio();
  CAMLreturn (Val_unit);
}

value sdlstub_unlock_audio(value u) {
  CAMLparam1(u);
  SDL_UnlockAudio();
  CAMLreturn (Val_unit);
}

value sdlstub_get_audio_status(value u) {
  CAMLparam1(u);
  CAMLreturn (Val_int(SDL_GetAudioStatus()));
}

value sdlstub_mix_audio(value b1, value b2, value volume) {
  CAMLparam3(b1, b2, volume);
  SDL_MixAudio(Data_bigarray_val(b1), Data_bigarray_val(b2), Bigarray_val(b1)->dim[0], Int_val(volume));
  CAMLreturn (Val_unit);
}

value sdlstub_convert_audio(value from_format, value from_channels, value from_freq,
value to_format, value to_channels, value to_freq, value buffer) {
  CAMLparam5(from_format, from_channels, from_freq, to_format, to_channels);
  CAMLxparam2(to_freq, buffer);
  SDL_AudioCVT wav_cvt;
  int wav_len = Bigarray_val(buffer)->dim[0];
  unsigned char *wav_buf = Data_bigarray_val(buffer);
if(!SDL_BuildAudioCVT(&wav_cvt, Int_val(from_format), Int_val(from_channels),
    Int_val(from_freq),
    Int_val(to_format), Int_val(to_channels), Int_val(to_freq))
){
    caml_failwith("Unable to carry out conversion");
}
wav_cvt.buf = malloc(wav_len * wav_cvt.len_mult);
wav_cvt.len = wav_len;
memcpy(wav_cvt.buf, wav_buf, wav_len);
SDL_ConvertAudio(&wav_cvt);
CAMLreturn (alloc_bigarray_dims(BIGARRAY_UINT8 | BIGARRAY_C_LAYOUT, 1,
wav_cvt.buf, wav_len * wav_cvt.len_mult));
}
value sdlstub_convert_audio_byte(value * argv, int n){
    return sdlstub_convert_audio(argv[0], argv[1], argv[2], argv[3], argv[4],
        argv[5], argv[6]);
}

/* Audio effects */

// Author: A. Umbach sdl@lokigames.com
int fxShift(double pitch, Uint8 *source, Uint8 *target, int len) {
    int i, j, k;
    double l;
    double pa = 0;
    double shift = pitch > 0 ? pitch : 0;
    len /= 4;
    for(i = 0; i < len; i++) { // LR pairs
        for(j = 0; j < 2; j++) { // channels
            pa = i * shift;
            k = (int) pa;
            l = pa - k;
            *(Sint16*) (target + 2 * j + 4 * i) +=
                ( *(Sint16*) (source + 2 * j + 4 * (k + 0) ) * ( 1 - l ) +
                *(Sint16*) (source + 2 * j + 4 * (k + 2) ) * ( 1 ) );
        }
    }
    return ( (int)(len * shift + 0.49999) ) * 4;
}

value fxstub_shift(value shift, value source, value target)
{
    CAMLparam3(shift, source, target);
    int consumed = 0;
    consumed = fxShift(Double_val(shift), Data_bigarray_val(source),
        Data_bigarray_val(target), Bigarray_val(target)->dim[0] );
    CAMLreturn (Val_int(consumed));
}

// Author: A. Umbach sdl@lokigames.com
void fxPan(double pan, double vol, Uint8 *buf, Uint8 *out, int len) {
    int i;
```c
double left_vol = -vol * (-1.0 + pan) / 2.0;
double right_vol = vol * (1.0 + pan) / 2.0;

// Guards
if(left_vol < 0) left_vol = 0;
if(left_vol > 1) left_vol = 1;
if(right_vol < 0) right_vol = 0;
if(right_vol > 1) right_vol = 1;

for(i = 0; i < len; i += 4) {
    *(Sint16*) (out + i) = *(Sint16*) (buf + i) * left_vol;
    *(Sint16*) (out + i + 2) = *(Sint16*) (buf + i + 2) * right_vol;
}

value fxstub_pan(value pan, value vol, value buf, value out)
{
    CAMLparam4(pan, vol, buf, out);
    fxPan(Double_val(pan), Double_val(vol), Data_bigarray_val(buf),
          Data_bigarray_val(out), Bigarray_val(buf)->dim[0]);
    CAMLreturn (Val_unit);
}

/* the following code is borrowed from stars.c by Nathan Strong */

value sdldraw_put_pixel(value s, value vx, value vy, value vc) {
    CAMLparam4(s, vx, vy, vc);
    Uint16 x = Int_val(vx);
    Uint16 y = Int_val(vy);
    Uint32 pixel = Int32_val(vc);
    SDL_Surface* dst = (SDL_Surface*) s;
    Uint8 *pix;
    int shift;

    if(SDL_LockSurface(dst) == 0) {
        switch (dst->format->BytesPerPixel) {
            case 1:
                *((Uint8*) dst->pixels + y * dst->pitch + x) = pixel;
                break;
            case 2:
                *((Uint16*) dst->pixels + y * dst->pitch/2 + x) = pixel;
                break;
            case 3:
                /* Slow, but endian correct */
                pix = (Uint8*) dst->pixels + y * dst->pitch + x*3;
                shift = dst->format->Rshift;
                *(pix+shift/8) = pixel>>shift;
                shift = dst->format->Gshift;
                *(pix+shift/8) = pixel>>shift;
                shift = dst->format->Bshift;
                *(pix+shift/8) = pixel>>shift;
                break;
        }
    }
}
case 4:
    *((Uint32 *)dst->pixels + y * dst->pitch/4 + x) = pixel;
    break;
default:
    break;
};
SDL_UnlockSurface(dst);
};
CAMLreturn(Val_unit);
}

value sdldraw_get_pixel(value s, value vx, value vy) {
    CAMLparam3(s,vx,vy);
    CAMLlocal1(rs);
    Uint16 x = Int_val(vx);
    Uint16 y = Int_val(vy);
    Uint32 pixel = 0;
    SDL_Surface* dst = (SDL_Surface*) s;
    Uint8 *pix;
    int shift;

    if (SDL_LockSurface(dst) == 0) {
        switch (dst->format->BytesPerPixel) {
        case 1:
            pixel = *((Uint8 *)dst->pixels + y * dst->pitch + x);
            break;
        case 2:
            pixel = *((Uint16 *)dst->pixels + y * dst->pitch/2 + x);
            break;
        case 3:
            /* Slow, but endian correct */
            pix = (Uint8 *)dst->pixels + y * dst->pitch + x*3;
            pixel = 0;
            shift = dst->format->Rshift;
            pixel |= (*(pix+shift/8)) << shift;
            shift = dst->format->Gshift;
            pixel |= (*(pix+shift/8)) << shift;
            shift = dst->format->Bshift;
            pixel |= (*(pix+shift/8)) << shift;
            break;
        case 4:
            pixel = *((Uint32 *)dst->pixels + y * dst->pitch/4 + x);
            break;
        default:
            break;
        }
        SDL_UnlockSurface(dst);
    };

    rs = caml_copy_int32(pixel);
    CAMLreturn(rs);
```c
#ifdef __APPLE__
int main(int argc, char **argv)
{
    caml_main(argv);
    return 0;
}
#endif

tt.ml

open Sdl

type action = Ast | Bytecode | Compile

let main() =
    let leave () =
        print_string ("Usage: " ^ Sys.argv.(0) ^ " <file.wav>");
        Sdl.quit ();
    ()
    in
    if Array.length Sys.argv < 2 then leave () else begin
        let lexbuf = Lexing.from_channel stdin in
        let program = Parser.program Scanner.token lexbuf in
        Execute.execute_prog (Compile.translate program) Sys.argv.(1)
    end

let _ = main ()

testall.sh

#!/bin/sh

TT=./tt

# Set time limit for all operations
ulimit -t 30

globallog=testall.log
rm -f $globallog
error=0
globalerror=0

keep=0

Usage() {
    echo "Usage: testall.sh [options] [.tt files]"
    echo "-k    Keep intermediate files"
    echo "-h    Print this help"
    exit 1
}
SignalError() {
    if [ $error -eq 0 ] ; then
        echo "FAILED"
        error=1
    fi
    echo "$1"
}

# Compare <outfile> <reffile> <difffile>
# Compares the outfile with reffile. Differences, if any, written to difffile
Compare() {
    generatedfiles="$generatedfiles $3"
    echo diff -b $1 $2 "$3" 1>&2
    diff -b "$1" "$2" "$3" 2>&1 || {
        SignalError "$1 differs"
        echo "FAILED $1 differs from $2" 1>&2
    }
}

# Run <args>
# Report the command, run it, and report any errors
Run() {
    echo $* 1>&2
    eval $* || {
        SignalError "$1 failed on $*"
        return 1
    }
}

Check() {
    error=0
    basename=`echo $1 | sed 's/.*\///'`
    reffile=`echo $1 | sed 's/.tt$//'`
    basedir="`echo $1 | sed 's/\[/\[^/\]*/\]/'`/"
    echo -n "$basename..."
    echo 1>&2
    echo "##### Testing $basename" 1>&2
    generatedfiles=""
    generatedfiles="$generatedfiles ${basename}.c.out" &&
    Run "$TT" "a.wav" "$1" "${basename}.c.out" &&
    Compare ${basename}.c.out ${reffile}.out ${basename}.c.diff
    # Report the status and clean up the generated files
    if [ $error -eq 0 ] ; then
        if [ $keep -eq 0 ] ; then
            rm -f $generatedfiles
        fi
    fi
}
```bash
# XXX: Before the rest of the script

echo "OK"

else
  echo "####### FAILED" 1>&2
globalerror=$error
fi

} while getopts kdpsh c; do
  case $c in
    k) # Keep intermediate files
      keep=1
      ;;
    h) # Help
      Usage
      ;;
    esac
done

shift `expr $OPTIND - 1`

if [ $# -ge 1 ];
then
  files=$@
else
  #files="test-suite/fail-*.* test-suite/test-*.*"
  files="test-suite/test-*.*"
fi

for file in $files
do
  case $file in
    *test-*)
      Check $file 2>> $globallog
      ;;
    *fail-*)
      CheckFail $file 2>> $globallog
      ;;
    *)
      echo "unknown file type $file"
globalerror=1
      ;;
  esac
done

exit $globalerror

Makefile

OBJS = ast.cmo parser.cmo scanner.cmo bytecode.cmo compile.cmo execute.cmo tt.cmo

TESTS = \
```
arith1
arith2
fib
for1
func1
func2
func3
gcd
global1
hello
if1
if2
if3
if4
ops1
var1
while1
 setPosition
 setColor
 setDiscoColor
 setLineWidth
 setDash
 setOrientation
 pd
 pu
 fd
 bk
 rt
 lt
 setRate
 play
 pause
 cue
 setVolume
 getTuneLength
 getSample
 setSample
 stack
 boundaries

TARFILES = Makefile testall.sh scanner.mll parser.mly
   ast.ml bytecode.ml compile.ml execute.ml tt.ml
   $(TESTS:%=test-suite/test-%.tt) \\
   $(TESTS:%=test-suite/test-%.out)

 tt : $(OBJS)
      ocamlc -custom -I lib/ -ccopt -Llib/ bigarray.cma lib/sdl_stub.o -cclib -LSDL -o tt lib/sdl.cmo $(OBJS)

.PHONY : test
test : tt testall.sh
Appendix B: Translation Examples

The PostScript listing of the Hi! (world) example presented first in the Section 2.1 tutorial. A second example output follows this listing on page 163.

stroke
newpath
0 24 moveto
stroke newpath
0 24 moveto
<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.77999999991</td>
<td>-0.5000000003195</td>
<td>rlineto</td>
</tr>
<tr>
<td>-1.78000000009</td>
<td>-0.499999996805</td>
<td>rlineto</td>
</tr>
<tr>
<td>2.3399999991</td>
<td>-0.50000000042</td>
<td>rlineto</td>
</tr>
<tr>
<td>-2.34000000009</td>
<td>-0.4999999958</td>
<td>rlineto</td>
</tr>
<tr>
<td>2.2099999991</td>
<td>-0.500000003967</td>
<td>rlineto</td>
</tr>
<tr>
<td>-2.21000000009</td>
<td>-0.499999996033</td>
<td>rlineto</td>
</tr>
<tr>
<td>2.2899999991</td>
<td>-0.50000000411</td>
<td>rlineto</td>
</tr>
<tr>
<td>2.3399999991</td>
<td>-0.50000000042</td>
<td>rlineto</td>
</tr>
<tr>
<td>-2.34000000009</td>
<td>-0.4999999958</td>
<td>rlineto</td>
</tr>
<tr>
<td>1.9899999991</td>
<td>-0.500000003572</td>
<td>rlineto</td>
</tr>
<tr>
<td>-1.99000000009</td>
<td>-0.499999996428</td>
<td>rlineto</td>
</tr>
<tr>
<td>1.9799999991</td>
<td>-0.500000003554</td>
<td>rlineto</td>
</tr>
<tr>
<td>-1.98000000009</td>
<td>-0.499999996446</td>
<td>rlineto</td>
</tr>
<tr>
<td>1.6099999991</td>
<td>-0.50000000289</td>
<td>rlineto</td>
</tr>
<tr>
<td>-1.61000000009</td>
<td>-0.49999999711</td>
<td>rlineto</td>
</tr>
<tr>
<td>1.6099999991</td>
<td>-0.50000000289</td>
<td>rlineto</td>
</tr>
<tr>
<td>-1.61000000009</td>
<td>-0.49999999711</td>
<td>rlineto</td>
</tr>
<tr>
<td>1.2799999991</td>
<td>-0.500000002297</td>
<td>rlineto</td>
</tr>
<tr>
<td>-1.28000000009</td>
<td>-0.499999997703</td>
<td>rlineto</td>
</tr>
<tr>
<td>1.2799999991</td>
<td>-0.500000002297</td>
<td>rlineto</td>
</tr>
<tr>
<td>-1.28000000009</td>
<td>-0.499999997703</td>
<td>rlineto</td>
</tr>
<tr>
<td>1.0299999991</td>
<td>-0.500000001849</td>
<td>rlineto</td>
</tr>
<tr>
<td>-1.03000000009</td>
<td>-0.499999998151</td>
<td>rlineto</td>
</tr>
<tr>
<td>0.969999999103</td>
<td>-0.500000001741</td>
<td>rlineto</td>
</tr>
<tr>
<td>-0.970000000897</td>
<td>-0.499999998259</td>
<td>rlineto</td>
</tr>
<tr>
<td>0.799999999103</td>
<td>-0.500000001436</td>
<td>rlineto</td>
</tr>
<tr>
<td>-0.800000000897</td>
<td>-0.499999998564</td>
<td>rlineto</td>
</tr>
<tr>
<td>0.739999999103</td>
<td>-0.500000001328</td>
<td>rlineto</td>
</tr>
<tr>
<td>-0.740000000897</td>
<td>-0.499999998672</td>
<td>rlineto</td>
</tr>
<tr>
<td>0.679999999103</td>
<td>-0.500000001221</td>
<td>rlineto</td>
</tr>
<tr>
<td>-0.680000000897</td>
<td>-0.499999998779</td>
<td>rlineto</td>
</tr>
<tr>
<td>0.609999999103</td>
<td>-0.500000001095</td>
<td>rlineto</td>
</tr>
<tr>
<td>-0.610000000897</td>
<td>-0.499999998905</td>
<td>rlineto</td>
</tr>
<tr>
<td>0.629999999103</td>
<td>-0.500000001131</td>
<td>rlineto</td>
</tr>
<tr>
<td>-0.630000000897</td>
<td>-0.499999998869</td>
<td>rlineto</td>
</tr>
<tr>
<td>0.529999999103</td>
<td>-0.500000000951</td>
<td>rlineto</td>
</tr>
<tr>
<td>-0.530000000897</td>
<td>-0.499999999049</td>
<td>rlineto</td>
</tr>
<tr>
<td>0.549999999103</td>
<td>-0.500000000987</td>
<td>rlineto</td>
</tr>
<tr>
<td>-0.550000000897</td>
<td>-0.499999999013</td>
<td>rlineto</td>
</tr>
<tr>
<td>0.469999999103</td>
<td>-0.500000000844</td>
<td>rlineto</td>
</tr>
<tr>
<td>-0.470000000897</td>
<td>-0.499999999156</td>
<td>rlineto</td>
</tr>
<tr>
<td>0.379999999103</td>
<td>-0.500000000682</td>
<td>rlineto</td>
</tr>
<tr>
<td>-0.380000000897</td>
<td>-0.499999999318</td>
<td>rlineto</td>
</tr>
<tr>
<td>0.299999999103</td>
<td>-0.500000000538</td>
<td>rlineto</td>
</tr>
<tr>
<td>-0.300000000897</td>
<td>-0.499999999462</td>
<td>rlineto</td>
</tr>
<tr>
<td>0.209999999103</td>
<td>-0.500000000377</td>
<td>rlineto</td>
</tr>
<tr>
<td>-0.210000000897</td>
<td>-0.499999999623</td>
<td>rlineto</td>
</tr>
</tbody>
</table>
The second example has Turtle dancing to his favorite trumpet in rows of Hi’s with shifting colors. The TT listing, resulting PostScript image, and PostScript listing follow.

/* hi4.tt */

int x;
int y;

routine main
{

stroke newpath
425 230 moveto
Showpage

"163"
int i;
int j;
int height;
height = 142;
x = 5;
y = 29;

pu~;
fd->5|;
lt->90|;
fd->5|;

play~;

for(i = 1; i<5; i = i + 1)
{
  while(x < (612 - ((height/4) + (height/6) + 5)))
  {
    setColor-> 30*i + x/3, 45*i + x/4, 50*i|;
    drawHi->height|;
    x = x + 80;
    setPosition-> x, y|;
  }
  x = 5;
  y = y + height + 20;
  setPosition-> 5, y|;
}
}

routine drawH h
{
  setLineWidth->1|;
  pd~;
  fd->h|;
  pu~;
  bk-> h/2 |;
  pd~;
  rt->90|;
  fd-> h/4 |;
  pu~;
  lt->90|;
  fd-> h/2 |;
  pd~;
  bk->h|;
}

routine drawi h
{
  pd~;
  fd-> h * 5 / 8 |;
  pu~;
  fd-> 2 * (h/100) + 4 |;
  setLineWidth-> 5|;
pd~;  
fd-> 2 * (h/100) + 4 |;  
pu~;  
}

routine drawHi h  
{  
drawH->h|;  
pu~;  
rt->90|;  
fhd-> h/6 |;  
lt->90|;  
drawi->h|;  
bk-> h * 5/ 8 + 4 * (h/100) + 8|;  
x = x + (h/4) + (h/6);  
}  

stroke  
newpath
-0.269999999103 0.500000000485 rlineto
0.2500000000897 0.49999999551 rlineto
-1.54999999901 0.5000000002782 rlineto
1.5600000009 0.4999999972 rlineto
-2.129999999103 0.500000003823 rlineto
2.1200000009 0.499999996195 rlineto
-0.849999999103 0.500000001526 rlineto
0.8500000000897 0.499999998474 rlineto
-0.189999999103 0.50000000341 rlineto
0.1800000000897 0.49999999677 rlineto
-1.4799999991 0.500000002656 rlineto
1.4900000009 0.499999997326 rlineto
-2.0299999991 0.500000003644 rlineto
2.0300000009 0.499999996356 rlineto
-0.749999999103 0.500000001346 rlineto
0.7500000000897 0.499999998654 rlineto
0.2100000000897 0.49999999623 rlineto
-0.209999999103 0.500000000377 rlineto
-1.1299999991 0.500000002028 rlineto
1.1400000009 0.499999997954 rlineto
-1.2099999991 0.500000002172 rlineto
1.2200000009 0.49999999781 rlineto
-0.0299999991026 0.500000000054 rlineto
0.03000000008974 0.49999999946 rlineto
-1.2199999991 0.50000000219 rlineto
1.2400000009 0.499999997774 rlineto
-0.119999999103 0.50000000215 rlineto
0.1200000000897 0.49999999785 rlineto
0.0200000000897 0.4999999964 rlineto
8.97448368483e-10 0.5 rlineto
-1.5599999991 0.50000000028 rlineto
1.5700000009 0.499999997182 rlineto
-0.229999999103 0.500000000413 rlineto
0.2500000000897 0.49999999551 rlineto
-1.8899999991 0.500000003392 rlineto
1.9000000009 0.49999999659 rlineto
-1.9899999991 0.500000003572 rlineto
2.0200000009 0.499999996374 rlineto
-1.2699999991 0.50000000228 rlineto
1.2700000009 0.49999999772 rlineto
-0.339999999103 0.50000000061 rlineto
0.3700000000897 0.49999999336 rlineto
0.08000000008974 0.49999999856 rlineto
-0.0799999991026 0.500000000144 rlineto
-0.749999999103 0.500000001346 rlineto
0.7900000000897 0.499999998582 rlineto
-0.589999999103 0.500000001059 rlineto
0.5900000000897 0.49999999841 rlineto
2.4699999991 -0.5000000004433 rlineto
-2.4800000009 -0.49999999549 rlineto
1.4399999991 -0.500000002585 rlineto
-1.4300000009 -0.49999997433 rlineto
1.3499999991 -0.500000002423 rlineto
-1.3500000009 -0.49999997577 rlineto
0.399999999103 -0.500000000718 rlineto
-0.390000000897 -0.499999993 rlineto
0.389999999103 -0.5000000007 rlineto
-0.390000000897 -0.499999993 rlineto
1.9599999991 -0.500000003518 rlineto
-1.9600000009 -0.49999996482 rlineto
2.0099999991 -0.500000003608 rlineto
-2.0100000009 -0.49999996392 rlineto
0.749999999103 -0.500000001346 rlineto
-0.740000000897 -0.49999998672 rlineto
0.789999999103 -0.500000001418 rlineto
-0.790000000897 -0.49999998582 rlineto
1.8199999991 -0.500000003267 rlineto
-1.8200000009 -0.49999996733 rlineto
1.8799999991 -0.500000003374 rlineto
-1.8800000009 -0.49999996626 rlineto
0.129999999103 -0.500000000233 rlineto
-0.120000000897 -0.4999999785 rlineto
0.199999999103 -0.500000000359 rlineto
-0.200000000897 -0.4999999641 rlineto
0.899999999103 -0.500000001615 rlineto
-0.900000000897 -0.49999998385 rlineto
0.969999999103 -0.500000001741 rlineto
-0.970000000897 -0.49999998259 rlineto
1.7799999991 -0.500000003195 rlineto
-1.7800000009 -0.49999996805 rlineto
1.7599999991 -0.500000003159 rlineto
-1.7600000009 -0.49999996841 rlineto
0.179999999103 -0.500000000323 rlineto
-0.170000000897 -0.49999999695 rlineto
stroke newpath
40 27 moveto
23. 0. rmoveto
stroke
newpath
63 27 moveto
-0.0999999991026 0.500000000179 rlineto
1.2500000009 0.49999997756 rlineto
-1.2499999991 0.500000002244 rlineto
1.1700000009 0.4999999979 rlineto
-1.1699999991 0.5000000021 rlineto
2.4100000009 0.49999995674 rlineto
-8.97448368483e-10  -0.5  rlineto
0.509999999103  -0.5000000000915  rlineto
-0.510000000897  -0.499999999085  rlineto
0.169999999103  -0.5000000000305  rlineto
-0.170000000897  -0.499999999695  rlineto
0.719999999103  -0.500000001292  rlineto
-0.720000000897  -0.499999998708  rlineto
0.349999999103  -0.500000000628  rlineto
-0.350000000897  -0.499999999372  rlineto
0.949999999103  -0.500000001705  rlineto
-0.950000000897  -0.499999998295  rlineto
0.579999999103  -0.500000001041  rlineto
-0.580000000897  -0.499999998959  rlineto
1.1999999991  -0.500000002154  rlineto
-1.20000000009  -0.499999997846  rlineto
0.909999999103  -0.500000001633  rlineto
-0.910000000897  -0.499999998367  rlineto
1.6599999991  -0.500000000298  rlineto
-1.66000000009  -0.49999999702  rlineto
1.3999999991  -0.500000002513  rlineto
-1.40000000009  -0.499999997487  rlineto
2.2999999991  -0.500000004128  rlineto
-2.30000000009  -0.499999995872  rlineto
2.0499999991  -0.500000000368  rlineto
stroke
newpath
451 25 moveto
23. 0. rmoveto
stroke
newpath
474 25 moveto
0.530000000897  0.499999999049  rlineto
-0.519999999103  0.500000000933  rlineto
0.310000000897  0.499999999444  rlineto
-0.309999999103  0.500000000556  rlineto
1.44000000009  0.499999997415  rlineto
-1.4399999991  0.500000002585  rlineto
1.26000000009  0.499999997738  rlineto
-1.2599999991  0.500000002262  rlineto
-0.0299999991026  0.500000000054  rlineto
0.0400000008974  0.49999999928  rlineto
2.40000000009  0.499999995692  rlineto
-2.4099999991  0.500000004326  rlineto
1.21000000009  0.499999997828  rlineto
-1.1999999991  0.500000002154  rlineto
1.11000000009  0.499999998008  rlineto
-1.1099999991  0.500000001992  rlineto
0.0100000008974  0.49999999982  rlineto
8.97448368483e-10  0.5  rlineto
2.20000000009 0.499999996051 rlineto
-1.7699999991 0.500000003177 rlineto
1.75000000009 0.49999999859 rlineto
-0.869999999103 0.500000001562 rlineto
0.850000000897 0.499999998474 rlineto
0.0800000008974 0.49999999856 rlineto
-0.909999999103 0.500000000144 rlineto
-0.259999999103 0.500000000467 rlineto
0.2400000000897 0.49999999569 rlineto
-1.7899999991 0.500000003213 rlineto
1.80000000009 0.499999996769 rlineto
-0.149999999103 0.500000000269 rlineto
0.1200000000897 0.49999999785 rlineto
-1.6099999991 0.500000000289 rlineto
1.6200000009 0.499999997092 rlineto
-0.699999999103 0.500000001256 rlineto
0.6700000000897 0.499999998797 rlineto
-2.0799999991 0.5000000003733 rlineto
2.09000000009 0.499999996249 rlineto
-2.0899999991 0.5000000003751 rlineto
2.06000000009 0.499999996303 rlineto
-0.779999999103 0.500000000114 rlineto
0.7800000000897 0.4999999986 rlineto
-1.7699999991 0.5000000003177 rlineto
1.74000000009 0.499999996877 rlineto
-0.339999999103 0.500000000061 rlineto
0.3400000000897 0.49999999939 rlineto
0.2500000000897 0.499999999551 rlineto
-0.289999999103 0.500000000521 rlineto
-0.779999999103 0.500000000114 rlineto
0.7900000000897 0.499999998582 rlineto
-1.1999999991 0.5000000002154 rlineto
1.16000000009 0.499999997918 rlineto
-2.0399999991 0.5000000003662 rlineto
2.05000000009 0.49999999632 rlineto
-0.809999999103 0.5000000001454 rlineto
0.7600000000897 0.499999998636 rlineto
stroke newpath
6 332 moveto
-1.27437668325e-07 -71. rmoveto
stroke
newpath
5 261 moveto
0.5 -1.41 rlineto
0.5 0.91 rlineto
0.5 -0.86 rlineto
-0.659999999103 0.5000000001185 rlineto
1.1600000009 0.499999997918 rlineto
-1.1599999991 0.5000000002082 rlineto
1.2600000009 0.499999997738 rlineto
-1.2599999991 0.5000000002262 rlineto
1.8000000009 0.499999996769 rlineto
-1.7999999991 0.5000000003231 rlineto
2.0300000009 0.499999996356 rlineto
-2.0299999991 0.5000000003644 rlineto
0.0700000008974 0.49999999874 rlineto
-0.0599999991026 0.500000000108 rlineto
0.490000000897 0.499999999121 rlineto
-0.489999999103 0.500000000879 rlineto
1.1900000009 0.499999997864 rlineto
-1.1899999991 0.5000000002136 rlineto
1.8400000009 0.499999996697 rlineto
-1.8399999991 0.5000000003303 rlineto
0.0700000008974 0.49999999874 rlineto
-0.0599999991026 0.500000000108 rlineto
0.960000000897 0.499999998777 rlineto
-0.9599999991 0.500000001723 rlineto
1.8500000009 0.499999996679 rlineto
-1.8499999991 0.500000003321 rlineto
0.320000000897 0.49999999426 rlineto
-0.309999999103 0.50000000556 rlineto
1.3200000009 0.499999997631 rlineto
-1.3199999991 0.500000002369 rlineto
2.4500000009 0.499999995603 rlineto
-2.4499999991 0.500000004397 rlineto
1.0100000009 0.499999998187 rlineto
-0.999999999103 0.500000001795 rlineto
1.9900000009 0.499999996428 rlineto
-1.9899999991 0.500000003572 rlineto
0.710000000897 0.499999998726 rlineto
-0.699999999103 0.500000001256 rlineto
1.4000000009 0.499999997487 rlineto
-1.3999999991 0.500000002513 rlineto
0.230000000897 0.499999995857 rlineto
-0.219999999103 0.50000000395 rlineto
0.500000000897 0.499999999103 rlineto
-0.499999999103 0.50000000897 rlineto
1.9900000009 0.499999996428 rlineto
-1.9899999991 0.500000003572 rlineto
1.6900000009 0.499999996967 rlineto
-1.6899999991 0.500000003033 rlineto
0.760000000897 0.49999999836 rlineto
-0.749999999103 0.500000001346 rlineto
2.3200000009 0.499999995836 rlineto
-1.85000000009 -0.499999996679 rlineto
1.339999999 -0.500000002405 rlineto
-1.3600000009 -0.49999997559 rlineto
0.689999999103 -0.500000001283 rlineto
-0.690000000897 -0.49999998762 rlineto
2.0499999991 -0.50000000368 rlineto
0.479999999103 -0.50000000149 rlineto
1.1500000009 -0.49999999736 rlineto
1.1499999991 -0.500000002064 rlineto
-0.260000000897 -0.49999999533 rlineto
0.229999999103 -0.500000000413 rlineto
-0.830000000897 -0.49999999851 rlineto
0.829999999103 -0.50000000149 rlineto
-0.390000000897 -0.499999993 rlineto
0.359999999103 -0.500000000646 rlineto
-0.940000000897 -0.499999998313 rlineto
0.939999999103 -0.500000001687 rlineto
-0.750000000897 -0.499999998654 rlineto
0.719999999103 -0.500000001292 rlineto
-1.1700000009 -0.4999999979 rlineto
1.1699999991 -0.5000000021 rlineto
-1.1300000009 -0.499999997972 rlineto
1.0999999991 -0.500000001974 rlineto
-1.2800000009 -0.499999997703 rlineto
1.2799999991 -0.500000002297 rlineto
-1.1900000009 -0.499999997864 rlineto
1.1599999991 -0.500000002082 rlineto
-1.1000000009 -0.499999998026 rlineto
1.0999999991 -0.500000001974 rlineto
-0.730000000897 -0.49999999869 rlineto
0.699999999103 -0.500000001256 rlineto
-0.480000000897 -0.499999999138 rlineto
0.479999999103 -0.500000000862 rlineto
-2.2500000009 -0.499999995961 rlineto
2.2299999991 -0.500000004003 rlineto
-1.8900000009 -0.49999999608 rlineto
1.8899999991 -0.500000003392 rlineto
-0.790000000897 -0.499999998582 rlineto
0.759999999103 -0.500000001364 rlineto
-0.240000000897 -0.49999999569 rlineto
0.239999999103 -0.50000000431 rlineto
-1.5700000009 -0.499999997182 rlineto
1.5499999991 -0.500000002782 rlineto
-0.830000000897 -0.49999999851 rlineto
0.829999999103 -0.50000000149 rlineto
-2.2700000009 -0.499999995926 rlineto
2.2499999991 -0.500000004039 rlineto
-1.4100000009 -0.499999997469 rlineto
1.4099999999 -0.500000002531 rlineto
-0.550000000897 -0.49999999013 rlineto
0.519999999103 -0.500000000933 rlineto
-2.1300000009 -0.49999996177 rlineto
2.1399999991 -0.500000003841 rlineto
-1.8200000009 -0.49999996733 rlineto
1.7899999991 -0.500000003213 rlineto
-0.720000000897 -0.49999998708 rlineto
0.719999999103 -0.500000001292 rlineto
-1.0600000009 -0.49999998097 rlineto
1.0299999991 -0.500000001849 rlineto
0.119999999103 -0.500000000215 rlineto
-0.120000000897 -0.4999999785 rlineto
-0.920000000897 -0.49999998349 rlineto
0.889999999103 -0.500000001597 rlineto
-2.1600000009 -0.49999996123 rlineto
2.1699999991 -0.500000003895 rlineto
-1.2500000009 -0.49999997756 rlineto
1.2099999991 -0.500000002172 rlineto
0.189999999103 -0.50000000341 rlineto
-0.190000000897 -0.49999999659 rlineto
-1.8000000009 -0.49999996769 rlineto
1.7699999991 -0.500000003177 rlineto
-0.260000000897 -0.49999999533 rlineto
0.259999999103 -0.500000000467 rlineto
0.279999999103 -0.500000000503 rlineto
-0.320000000897 -0.49999999426 rlineto
-0.640000000897 -0.49999998851 rlineto
0.649999999103 -0.500000001167 rlineto
0.169999999103 -0.500000000305 rlineto
-0.210000000897 -0.49999999623 rlineto
-0.650000000897 -0.49999998833 rlineto
0.659999999103 -0.500000001185 rlineto
-1.7700000009 -0.49999996823 rlineto
1.7399999991 -0.500000003123 rlineto
0.0299999991026 -0.500000000054 rlineto
-0.0300000008974 -0.49999999946 rlineto
-0.0900000008974 -0.49999999838 rlineto
0.0599999991026 -0.500000000108 rlineto
-0.830000000897 -0.4999999851 rlineto
0.839999999103 -0.500000001508 rlineto
0.229999999103 -0.500000000413 rlineto
-0.260000000897 -0.49999999533 rlineto
-0.520000000897 -0.49999999067 rlineto
0.529999999103 -0.500000000951 rlineto
-0.680000000897 -0.49999998779 rlineto
0.659999999103 -0.500000001185 rlineto
-1.4300000009 -0.49999997433 rlineto
-0.310000000897 -0.499999999444 rlineto
0.309999999103 -0.500000000556 rlineto
-1.8300000009 -0.499999996715 rlineto
1.8099999991 -0.500000003249 rlineto
-1.5000000009 -0.499999997308 rlineto
1.4999999991 -0.500000002692 rlineto
-0.0100000008974 -0.49999999982 rlineto
-0.0200000008974 -0.49999999964 rlineto
-2.1400000009 -0.499999996159 rlineto
2.1499999991 -0.500000003859 rlineto
-0.240000000897 -0.499999999569 rlineto
0.209999999103 -0.500000000377 rlineto
-0.310000000897 -0.499999999569 rlineto
0.209999999103 -0.500000000377 rlineto
0.0699999991026 -0.50000000126 rlineto
-0.0900000008974 -0.499999999838 rlineto
-1.8400000009 -0.49999999697 rlineto
1.8499999991 -0.500000003321 rlineto
-1.5200000009 -0.499999997272 rlineto
1.4999999991 -0.500000002692 rlineto
-0.820000000897 -0.499999998528 rlineto
0.819999999103 -0.500000001472 rlineto
0.129999999103 -0.50000000233 rlineto
-0.150000000897 -0.49999999731 rlineto
-1.6500000009 -0.499999997038 rlineto
1.6599999991 -0.50000000298 rlineto
0.0399999991026 -0.50000000072 rlineto
-0.0600000008974 -0.49999999892 rlineto
-1.7700000009 -0.49999996823 rlineto
1.7799999991 -0.500000003195 rlineto
-1.8300000009 -0.499999996715 rlineto
1.8199999991 -0.500000003267 rlineto
-1.0500000009 -0.499999998115 rlineto
1.0499999991 -0.500000001885 rlineto
-0.250000000897 -0.49999999551 rlineto
0.239999999103 -0.50000000431 rlineto
-2.0400000009 -0.499999996338 rlineto
2.0499999991 -0.50000000368 rlineto
-0.390000000897 -0.4999999993 rlineto
0.379999999103 -0.50000000682 rlineto
0.409999999103 -0.50000000736 rlineto
-0.410000000897 -0.49999999264 rlineto
0.259999999103 -0.50000000467 rlineto
-0.260000000897 -0.49999999533 rlineto
-1.5000000009 -0.499999997308 rlineto
1.5099999991 -0.50000000271 rlineto
-1.0800000009 -0.49999998062 rlineto
1.0799999991 -0.500000001938 rlineto
1.0100000009 0.499999998187 rlineto
-2.0499999991 0.50000000368 rlineto
2.0500000009 0.49999999632 rlineto
-1.7799999991 0.500000003195 rlineto
1.7900000009 0.49999996787 rlineto
-0.2299999991 0.500000000413 rlineto
0.220000000897 0.49999999605 rlineto
-0.9399999991 0.500000001687 rlineto
0.960000000897 0.49999998277 rlineto
-1.9299999991 0.500000003464 rlineto
1.9300000009 0.49999996536 rlineto
-1.7799999991 0.500000003195 rlineto
1.8000000009 0.49999996769 rlineto
-0.079999999102 0.500000000144 rlineto
0.0700000008974 0.4999999874 rlineto
-2.2299999991 0.500000004003 rlineto
2.2600000009 0.4999999544 rlineto
-0.339999999103 0.50000000061 rlineto
0.330000000897 0.49999999408 rlineto
-0.189999999103 0.50000000341 rlineto
-2.3499999991 0.500000004218 rlineto
1.8400000009 0.49999996697 rlineto
-1.8399999991 0.500000003303 rlineto
1.3900000009 0.49999997505 rlineto
-1.3699999991 0.500000002459 rlineto
1.0200000009 0.49999998169 rlineto
-1.0199999991 0.500000001831 rlineto
2.3400000009 0.499999958 rlineto
-2.3299999991 0.500000004182 rlineto
2.2100000009 0.49999996033 rlineto
-2.2099999991 0.500000003967 rlineto
0.0500000008974 0.4999999999 rlineto
-0.0299999991026 0.500000000054 rlineto
0.200000000897 0.49999999641 rlineto
-0.199999999103 0.500000000359 rlineto
2.1800000009 0.49999996087 rlineto
-2.1799999991 0.500000003913 rlineto
0.0100000008974 0.4999999982 rlineto
8.97448368483e-10 0.5 rlineto
1.1200000009 0.4999999799 rlineto
-1.1199999991 0.500000003201 rlineto
1.6800000009 0.4999999685 rlineto
-1.6799999991 0.500000002315 rlineto
1.9900000009 0.49999996428 rlineto
-1.9899999991 0.500000003572 rlineto
0.200000000897 0.49999999641 rlineto
-0.189999999103 0.500000000341 rlineto
2.3200000009 0.49999995836 rlineto
0.5 -0.53 rlineto
0.5 1.34 rlineto
0.5 -1.35 rlineto
0.5 0. rlineto
stroke newpath
38 423 moveto
1.27437668325e-07 71. rmoveto
stroke
newpath
38 494 moveto
-0.670000000897 -0.499999998797 rlineto
0.679999999103 -0.500000001221 rlineto
-0.300000000897 -0.499999999462 rlineto
0.249999999103 -0.500000000449 rlineto
-0.900000000897 -0.499999998385 rlineto
0.909999999103 -0.500000001633 rlineto
-1.5300000009 -0.499999997254 rlineto
1.4799999991 -0.500000002656 rlineto
0.529999999103 -0.500000000951 rlineto
-0.530000000897 -0.499999999049 rlineto
-0.770000000897 -0.499999998618 rlineto
0.719999999103 -0.500000001292 rlineto
-1.1300000009 -0.499999997972 rlineto
1.1399999991 -0.500000002046 rlineto
0.049999999103 -0.500000000209 rlineto
-0.110000000897 -0.49999999803 rlineto
-0.160000000897 -0.49999999713 rlineto
0.169999999103 -0.50000000305 rlineto
-0.390000000897 -0.4999999993 rlineto
0.339999999103 -0.50000000061 rlineto
-0.520000000897 -0.49999999067 rlineto
0.529999999103 -0.500000000951 rlineto
-0.530000000897 -0.499999999049 rlineto
0.489999999103 -0.500000000879 rlineto
-0.630000000897 -0.49999998869 rlineto
0.639999999103 -0.500000001149 rlineto
-0.940000000897 -0.49999998313 rlineto
0.919999999103 -0.500000001651 rlineto
-1.0700000009 -0.49999998079 rlineto
1.0799999991 -0.500000001938 rlineto
0.409999999103 -0.500000000736 rlineto
-0.410000000897 -0.49999999264 rlineto
0.149999999103 -0.500000000269 rlineto
-0.140000000897 -0.49999999749 rlineto
0.109999999103 -0.500000000197 rlineto
-0.070000000897 -0.4999999874 rlineto
-0.480000000897 -0.49999999138 rlineto
0.489999999103 -0.500000000879 rlineto
stroke
newpath
59 445 moveto
1.4100000009 0.499999997469 rlineto
-1.3999999991 0.500000002513 rlineto
1.7500000009 0.49999996859 rlineto
-1.7599999991 0.500000003159 rlineto
-0.0399999991426 0.50000000072 rlineto
0.06000000008974 0.49999999892 rlineto
0.2200000000897 0.49999999605 rlineto
-0.229999999103 0.50000000413 rlineto
0.9800000000897 0.499999998241 rlineto
-0.969999999103 0.500000001741 rlineto
1.2500000009 0.49999997756 rlineto
-1.2599999991 0.500000002262 rlineto
stroke newpath
59 450 moveto
-1.79489673697e-07 -100. rmoveto
143 353 moveto
0.537254901961 0.666666666667 0.588235294118 setrgbcolor
1 setlinewidth
stroke
newpath
143 353 moveto
-2.0099999991 0.500000003608 rlineto
2.3100000009 0.499999995854 rlineto
-2.3199999991 0.500000004164 rlineto
0.6300000000897 0.49999999869 rlineto
-0.609999999103 0.500000001095 rlineto
0.9700000000897 0.499999998259 rlineto
-0.979999999103 0.500000001759 rlineto
1.9800000009 0.499999996446 rlineto
-1.9699999991 0.500000003536 rlineto
2.3800000009 0.499999995728 rlineto
-2.3899999991 0.50000000429 rlineto
1.0100000009 0.499999998187 rlineto
-0.989999999103 0.500000001777 rlineto
1.4700000009 0.499999997362 rlineto
-1.4799999991 0.500000002656 rlineto
0.3500000000897 0.49999999372 rlineto
-0.329999999103 0.500000000592 rlineto
0.8400000000897 0.499999998492 rlineto
-0.849999999103 0.500000001526 rlineto
-0.0899999991026 0.500000000162 rlineto
0.1100000000897 0.49999999803 rlineto
0.4500000000897 0.49999999192 rlineto
-0.459999999103 0.50000000826 rlineto
2.1300000009 0.499999996177 rlineto
stroke newpath
174 422 moveto
1.27437668325e-07 71. rmoveto
stroke
newpath
174 493 moveto
0.509999999103 -0.500000000915 rlineto
-2.4200000009 -0.49999995656 rlineto
2.4199999991 -0.500000004344 rlineto
-0.48000000897 -0.49999999138 rlineto
0.479999999103 -0.50000000862 rlineto
-2.2000000009 -0.49999996051 rlineto
2.1999999991 -0.500000003949 rlineto
-1.2000000009 -0.49999997846 rlineto
1.2099999991 -0.500000002172 rlineto
-0.240000000897 -0.49999999569 rlineto
0.229999999103 -0.50000000413 rlineto
-0.730000000897 -0.4999999869 rlineto
0.749999999103 -0.50000001346 rlineto
-2.3100000009 -0.49999995854 rlineto
2.3099999991 -0.50000004146 rlineto
-2.3400000009 -0.499999958 rlineto
2.3599999991 -0.50000004236 rlineto
-1.2700000009 -0.4999999772 rlineto
1.2599999991 -0.50000002262 rlineto
-1.2200000009 -0.4999999781 rlineto
1.2399999991 -0.50000002226 rlineto
-0.0900000008974 -0.49999998388 rlineto
0.079999991026 -0.5000000144 rlineto
-0.200000000897 -0.4999999641 rlineto
0.219999999103 -0.5000000395 rlineto
-1.5900000009 -0.49999997146 rlineto
1.5899999991 -0.50000002854 rlineto
-1.8800000009 -0.49999996262 rlineto
-0.660000000897 -0.49999998815 rlineto
1.8899999991 -0.50000003392 rlineto
-1.9000000009 -0.4999999659 rlineto
1.4799999991 -0.50000002656 rlineto
-1.4600000009 -0.49999997379 rlineto
0.259999999103 -0.5000000467 rlineto
-0.260000000897 -0.4999999533 rlineto
2.1999999991 -0.50000003949 rlineto
-2.1900000009 -0.49999996069 rlineto
1.0999999991 -0.50000001974 rlineto
-1.1000000009 -0.49999998026 rlineto
0.279999999103 -0.5000000503 rlineto
-0.260000000897 -0.4999999533 rlineto
1.8399999991 -0.50000003303 rlineto
336
1.5100000009 0.49999999729 rlineto
-1.4299999991 0.500000002567 rlineto
-0.570000000897 0.49999999877 rlineto
0.950000000897 0.499999998295 rlineto
-0.899999999103 0.500000001615 rlineto
1.4300000009 0.499999997433 rlineto
-1.4499999991 0.500000002603 rlineto
0.170000000897 0.499999999695 rlineto
-0.129999999103 0.500000000682 rlineto
0.580000000897 0.499999998959 rlineto
-0.599999999103 0.500000001077 rlineto
-0.0599999991026 0.500000000108 rlineto
0.0800000008974 0.499999999856 rlineto
0.360000000897 0.499999999354 rlineto
-0.379999999103 0.500000000682 rlineto
0.700000000897 0.499999998744 rlineto
-0.699999999103 0.500000001256 rlineto
1.2000000009 0.49999999846 rlineto
-1.2199999991 0.50000000219 rlineto
0.280000000897 0.499999999497 rlineto
-0.289999999103 0.500000000521 rlineto
0.950000000897 0.499999998295 rlineto
-0.969999999103 0.500000001741 rlineto
-0.519999999103 0.500000000933 rlineto
0.500000000897 0.499999999103 rlineto
0.310000000897 0.499999999444 rlineto
-0.329999999103 0.500000000592 rlineto
1.6900000009 0.499999996967 rlineto
-1.7299999991 0.500000003105 rlineto
0.100000000897 0.499999999821 rlineto
-0.109999999103 0.500000000197 rlineto
0.100000000897 0.49999999821 rlineto
-0.139999999103 0.500000000251 rlineto
1.2500000009 0.499999997756 rlineto
-1.2699999991 0.50000000228 rlineto
0.550000000897 0.499999999013 rlineto
-0.589999999103 0.500000001059 rlineto
1.9600000009 0.499999996482 rlineto
-1.9799999991 0.500000003554 rlineto
0.900000000897 0.499999998385 rlineto
-0.939999999103 0.500000001687 rlineto
0.100000000897 0.49999999821 rlineto
-0.109999999103 0.500000000197 rlineto
1.3400000009 0.499999997595 rlineto
-1.3899999991 0.500000002495 rlineto
0.820000000897 0.499999998528 rlineto
-0.829999999103 0.50000000149 rlineto
23. 0. rmoveto
stroke
newpath
334 353 moveto
1.7800000009 0.499999996805 rlineto
-0.459999999103 0.5000000000826 rlineto
0.4600000000897 0.499999999174 rlineto
-2.3499999991 0.500000004218 rlineto
2.3200000009 0.499999995836 rlineto
-0.619999999103 0.500000001113 rlineto
0.620000000897 0.499999998887 rlineto
0.0100000008974 0.499999999982 rlineto
-0.0499999991026 0.50000000009 rlineto
-0.389999999103 0.50000000007 rlineto
0.4000000000897 0.49999999282 rlineto
-2.0999999991 0.500000003769 rlineto
2.0700000009 0.499999996285 rlineto
-2.1199999991 0.500000003805 rlineto
2.1300000009 0.499999996177 rlineto
-0.889999999103 0.500000001597 rlineto
0.8500000000897 0.49999999474 rlineto
-0.519999999103 0.500000000933 rlineto
0.5300000000897 0.49999999049 rlineto
-1.4899999991 0.500000002674 rlineto
1.4600000009 0.499999997379 rlineto
-0.799999999103 0.500000001436 rlineto
0.8100000000897 0.49999999546 rlineto
-1.4699999991 0.500000002638 rlineto
1.4400000009 0.499999997415 rlineto
-0.519999999103 0.500000000933 rlineto
0.5300000000897 0.49999999049 rlineto
-0.999999999103 0.500000001795 rlineto
0.9700000000897 0.499999998259 rlineto
0.1600000000897 0.49999999713 rlineto
-0.149999999103 0.500000000269 rlineto
-0.339999999103 0.50000000061 rlineto
0.3100000000897 0.49999999444 rlineto
-1.5399999991 0.500000002764 rlineto
1.5600000009 0.4999999972 rlineto
-2.2499999991 0.500000004039 rlineto
2.2200000009 0.499999996015 rlineto
-0.739999999103 0.500000001328 rlineto
0.7500000000897 0.499999998654 rlineto
-1.7699999991 0.500000003177 rlineto
1.7400000009 0.499999996877 rlineto
-0.109999999103 0.500000000197 rlineto
0.1200000000897 0.49999999785 rlineto
-1.5199999991 0.500000002728 rlineto
360
2.0999999991 -0.500000003769 rlineto
-2.1200000009 -0.499999996195 rlineto
2.1199999991 -0.500000003805 rlineto
-2.1600000009 -0.499999996123 rlineto
0.909999999103 -0.500000001633 rlineto
-0.920000000897 -0.499999998349 rlineto
1.1199999991 -0.50000000201 rlineto
-1.1600000009 -0.499999997918 rlineto
-0.0400000008974 -0.499999999288 rlineto
0.0299999991026 -0.500000000054 rlineto
0.749999999103 -0.500000001346 rlineto
-0.790000000897 -0.499999998582 rlineto
2.2699999991 -0.500000004074 rlineto
-2.2900000009 -0.49999999589 rlineto
1.5099999991 -0.50000000271 rlineto
-1.5400000009 -0.499999997236 rlineto
0.459999999103 -0.500000000826 rlineto
-0.470000000897 -0.499999999156 rlineto
1.0099999991 -0.500000001813 rlineto
-1.0400000009 -0.499999998133 rlineto
-0.0400000008974 -0.499999999288 rlineto
0.0299999991026 -0.500000000054 rlineto
1.9099999991 -0.500000003428 rlineto
-1.9400000009 -0.499999996518 rlineto
0.889999999103 -0.500000001597 rlineto
-0.900000000897 -0.499999998385 rlineto
1.6899999991 -0.500000003033 rlineto
-1.7100000009 -0.499999996931 rlineto
0.759999999103 -0.500000001364 rlineto
-0.770000000897 -0.499999998618 rlineto
0.429999999103 -0.500000000772 rlineto
-0.440000000897 -0.49999999921 rlineto
2.0899999991 -0.500000003751 rlineto
-2.1100000009 -0.499999996213 rlineto
0.689999999103 -0.500000001238 rlineto
-0.690000000897 -0.499999998762 rlineto
2.3599999991 -0.500000004236 rlineto
0.179999999103 -0.500000000323 rlineto
-0.160000000897 -0.499999999713 rlineto
0.159999999103 -0.500000000287 rlineto
-0.990000000897 -0.499999998223 rlineto
0.979999999103 -0.500000001759 rlineto
-2.1800000009 -0.499999996087 rlineto
2.1899999991 -0.500000003931 rlineto
-0.340000000897 -0.49999999939 rlineto
0.319999999103 -0.500000000574 rlineto
-0.480000000897 -0.499999999138 rlineto
0.489999999103 -0.500000000879 rlineto

408
-1.0500000009 -0.499999998115 rlineto
1.039999999 -0.500000001867 rlineto
-0.390000000897 -0.499999999 rlineto
0.399999999103 -0.500000000718 rlineto
-0.780000000897 -0.4999999986 rlineto
0.769999999103 -0.500000001382 rlineto
-2.0500000009 -0.49999999632 rlineto
2.0699999991 -0.500000003715 rlineto
-2.3500000009 -0.499999995782 rlinet
2.3399999991 -0.5000000042 rlineto
-0.440000000897 -0.4999999921 rlineto
0.449999999103 -0.50000000808 rlineto
-0.760000000897 -0.49999998636 rlineto
0.749999999103 -0.50000001346 rlineto
-0.790000000897 -0.49999998582 rlineto
0.809999999103 -0.50000001454 rlineto
-1.1700000009 -0.499999979 rlineto
1.1599999991 -0.50000002082 rlineto
-0.560000000897 -0.49999998995 rlineto
-1.9800000009 -0.49999996446 rlineto
1.3799999991 -0.50000002477 rlineto
1.1699999991 -0.500000021 rlineto
-2.4900000009 -0.4999999531 rlineto
0.0400000008974 -0.4999999928 rlineto
1.7799999991 -0.50000003195 rlineto
-1.8000000009 -0.49999996769 rlineto
0.889999999103 -0.50000001597 rlineto
-0.860000000897 -0.49999998456 rlineto
2.4399999991 -0.5000000438 rlineto
-2.4600000009 -0.49999995585 rlineto
1.8199999991 -0.50000003267 rlineto
-1.7900000009 -0.49999996787 rlineto
0.679999999103 -0.50000001221 rlineto
-0.690000000897 -0.49999998762 rlineto
0.0799999991026 -0.5000000144 rlineto
-0.0500000008974 -0.4999999991 rlineto
1.4499999991 -0.50000002603 rlineto
-1.4700000009 -0.49999997362 rlineto
0.909999999103 -0.50000001633 rlineto
-0.880000000897 -0.4999999842 rlineto
2.1999999991 -0.50000003949 rlineto
-2.2200000009 -0.49999996015 rlineto
1.6799999991 -0.50000003015 rlineto
-1.6500000009 -0.49999997038 rlineto
0.439999999103 -0.5000000079 rlineto
-0.450000000897 -0.49999999192 rlineto
2.4399999991 -0.5000000438 rlineto
-2.4200000009 -0.49999995656 rlineto
-1.6500000009 -0.499999997038 rlineto
0.809999999103 -0.500000001454 rlineto
-0.730000000897 -0.49999999869 rlineto
1.3099999991 -0.500000002351 rlineto
-1.4000000009 -0.499999997487 rlineto
1.7599999991 -0.500000003159 rlineto
-1.6800000009 -0.499999996985 rlineto
1.0299999991 -0.500000001849 rlineto
-1.1200000009 -0.49999999799 rlineto
2.2399999991 -0.500000004021 rlineto
-2.1600000009 -0.499999996123 rlineto
0.639999999103 -0.500000001149 rlineto
-0.730000000897 -0.49999999869 rlineto
2.2999999991 -0.500000004128 rlineto
-2.2200000009 -0.499999996015 rlineto
-0.330000000897 -0.499999999408 rlineto
1.7799999991 -0.500000003195 rlineto
-1.7000000009 -0.499999996949 rlineto
-0.0500000008974 -0.4999999991 rlineto
2.5199999991 -0.500000004523 rlineto
-1.7500000009 -0.499999996859 rlineto
-0.730000000897 -0.49999999869 rlineto
2.4099999991 -0.500000004326 rlineto
0.0499999991026 -0.50000000009 rlineto
-0.550000000897 -0.499999999013 rlineto
-1.9300000009 -0.499999996536 rlineto
0.00999999910255 -0.500000000018 rlineto
2.4699999991 -0.500000004433 rlineto
-2.1300000009 -0.499999996177 rlineto
-0.360000000897 -0.49999999354 rlineto
0.579999999103 -0.500000001041 rlineto
1.8999999991 -0.50000000341 rlineto
-1.1800000009 -0.499999997882 rlineto
-1.3200000009 -0.499999997631 rlineto
1.5499999991 -0.500000002782 rlineto
0.939999999103 -0.500000001687 rlineto
-0.0800000008974 -0.49999999856 rlineto
-2.4300000009 -0.499999995638 rlineto
0.309999999103 -0.500000000556 rlineto
2.1999999991 -0.500000003949 rlineto
-1.2700000009 -0.49999999772 rlineto
1.3099999991 -0.500000002351 rlineto
-0.740000000897 -0.499999998672 rlineto
0.689999999103 -0.500000001238 rlineto
-2.2100000009 -0.499999996033 rlineto
2.2499999991 -0.500000004039 rlineto
-1.7800000009 -0.499999996805 rlineto
1.7399999991 -0.500000003123 rlineto
-0.430000000897 -0.49999999228 rlineto
0.449999999103 -0.500000000808 rlineto
-0.400000000897 -0.49999999282 rlineto
0.369999999103 -0.500000000664 rlineto
-1.0500000009 -0.499999998115 rlineto
1.0699999991 -0.500000001921 rlineto
-1.8600000009 -0.499999996661 rlineto
1.8399999991 -0.500000003303 rlineto
-1.6000000009 -0.499999997128 rlineto
1.6099999991 -0.50000000289 rlineto
-0.920000000897 -0.499999998349 rlineto
0.899999999103 -0.500000001615 rlineto
-2.0800000009 -0.499999996267 rlineto
2.0899999991 -0.500000003751 rlineto
-0.0400000008974 -0.49999999928 rlineto
0.0199999991026 -0.50000000036 rlineto
-2.4100000009 -0.499999995674 rlineto
2.4199999991 -0.500000004344 rlineto
-1.6300000009 -0.499999997074 rlineto
1.6199999991 -0.500000002908 rlineto
-2.4700000009 -0.499999995567 rlineto
2.4699999991 -0.500000004433 rlineto
-0.370000000897 -0.49999999336 rlineto
0.359999999103 -0.500000000646 rlineto
-2.1800000009 -0.499999996087 rlineto
2.1799999991 -0.500000003913 rlineto
-1.2600000009 -0.499999997738 rlineto
1.2599999991 -0.500000002262 rlineto
-1.4400000009 -0.499999997415 rlineto
1.4299999991 -0.500000002567 rlineto
-1.5900000009 -0.499999997146 rlineto
1.5999999991 -0.500000002872 rlineto
-0.120000000897 -0.49999999785 rlineto
0.0999999991026 -0.500000000179 rlineto
-1.1700000009 -0.4999999979 rlineto
1.1899999991 -0.500000002136 rlineto
-0.680000000897 -0.499999998779 rlineto
0.659999999103 -0.500000001185 rlineto
0.0999999991026 -0.500000000179 rlineto
-0.0800000008974 -0.49999999856 rlineto
-0.240000000897 -0.49999999569 rlineto
0.219999999103 -0.50000000395 rlineto
-0.200000000897 -0.49999999641 rlineto
0.229999999103 -0.50000000413 rlineto
-1.3700000009 -0.499999997541 rlineto
1.3499999991 -0.500000002423 rlineto
-2.1500000009 -0.499999996141 rlineto
419 515 moveto
0.2800000000897 0.499999999497 rlineto
-1.9299999991 0.5000000003464 rlineto
1.9400000009 0.499999996518 rlineto
-1.1599999991 0.500000002082 rlineto
1.1400000009 0.49999997954 rlineto
0.4700000000897 0.49999999156 rlineto
-0.479999999103 0.500000000862 rlineto
-1.1799999991 0.50000002118 rlineto
1.1700000009 0.499999979 rlineto
-1.8099999991 0.50000003249 rlineto
1.8000000009 0.49999996769 rlineto
-0.719999999103 0.500000001292 rlineto
0.7000000000897 0.49999998744 rlineto
-0.929999999103 0.50000001669 rlineto
0.9100000000897 0.49999998367 rlineto
0.3000000000897 0.49999999462 rlineto
-0.319999999103 0.50000000574 rlineto
-0.969999999103 0.50000001741 rlineto
0.9500000000897 0.49999998295 rlineto
0.3000000000897 0.49999999462 rlineto
-0.319999999103 0.50000000574 rlineto
-0.179999999103 0.50000000323 rlineto
0.1600000000897 0.49999999713 rlineto
-1.4999999991 0.50000002692 rlineto
1.4900000009 0.49999997326 rlineto
-1.1799999991 0.50000002118 rlineto
1.1700000009 0.499999979 rlineto
0.0500000000897 0.4999999991 rlineto
-0.0699999991026 0.50000000126 rlineto
-0.919999999103 0.50000001651 rlineto
0.9400000000897 0.49999998313 rlineto
0.1200000000897 0.49999999785 rlineto
-0.139999999103 0.50000000251 rlineto
-1.8099999991 0.50000003249 rlineto
1.87000000009 0.49999996644 rlineto
-1.2299999991 0.50000002208 rlineto
1.2100000009 0.49999997828 rlineto
0.4300000000897 0.49999999228 rlineto
-0.339999999103 0.5000000061 rlineto
0.3200000000897 0.49999999426 rlineto
-0.339999999103 0.5000000061 rlineto
-1.8499999991 0.50000003321 rlineto
2.0000000009 0.4999999641 rlineto
-0.299999999103 0.50000000538 rlineto
0.2700000000897 0.49999999515 rlineto
-0.119999999103 0.50000000215 rlineto
0.3000000000897 0.49999999462 rlineto