MELODY

TONG GE
JINGSI LI
SHUO YANG

OVERVIEW

- Music programming language
- .mc → .csv → .midi
- Pitch → Note → Bar → Track → Melody
- Pitch & Rhythm → Bar → Track → Melody
- *~Cb4,(~G;8), [-r 4,4], [-b note1,note2], [-t bar1, bar2]*
- Concatenate (+), Append (←), Synthesize (&)
- Built-in functions, Self-defined functions
- Primitive types, Binop, Control-flow

TUTORIAL func_decl

- Melody program = main() + declarations
- It starts from main() and ends as main() ends
- Function declaration

```
function melody main(){
        …
        return melody_var;
}
function return_type func_name(para1_type para1,…){
        statement1;
        statement2;
        …
}
```

- bar at（track track1, int i)
  note at(bar bar1, int i)
- pitch/note/bar/track toneUp(pitch p/note n/bar b/track t, int i)
  pitch/note/bar/track toneDown(pitch p/note n/bar b/track t, int i)
- int length(bar b)
  int length(track t)

TUTORIAL built-in_func

- Variables should be declared before being assigned
- Variable declaration

  type var_name;

  track<<instrument, fraction, beats_per_bar, beats_per_minute, volume>> var_name;

  track<<>> var_name;

TUTORIAL var_decl

- Assignment should be within functions

Variable Type	Assignment Sample
int	*i=5;*
string	*s="Hello Melody!";*
bool	*b=true;*
pitch	*pitch1= ~C#6;*
note	*note1=(pitch1;4);*
rhythm	*rhythm1=[-r 4,4];*
bar	*bar1=[-b note1, (pitch1;4),(~C;4)]*
	bar2=[-b rhythm1;(pitch1,~D)]
track	*track1=[-t bar2,bar3]*
melody	*melody1=track1&track2*

TUTORIAL var_assign

- Concatenate
  track1=track2+track3;
- Appending
  bar1=bar2←(~A#;2);
  track1=track1←bar1;
- Synthesize
  melody1=track1&track2&track3;

TUTORIAL music_compo

Binop	Sample
+	*int3=int1+int2; string3=string1+string2;*
*	*int3=int1*int2; note2=note1*int1;*
==	*bool1=(int1==int2);, string, pitch, note*
!=	*bool1=(int1!=int2) ;,string, pitch, note*
<	*boo1=(int1<int2);*
<=	*boo1=(int1<=int2);*
>	*boo1=(int1>int2);*
>=	*boo1=(int1>=int2);*
&&	*bool1=bool2&&bool3;*
||	*bool1=bool2 | |bool3;*

TUTORIAL binary_op

if else	for	while

if (expression){
statement1;
statement2;
…
}
else{
statement3;
…
}
	for(i=0;i<10;i=i+1)
	{
	statement1;
	statement2;
	statement3;
	…
	}
		while(condition){
		statement1;
		statement2;
		…
		}

TUTORIAL ctrl_flow

```
/* the program starts from the main() function, and it ends when the main function ends */
function melody main(){
    bar b1; /* declare a bar variable */
    /* assign value to the bar variable with [-b...] with notes*/
    b1=[-b (~C6;1),(~C6;1),(~G6;1),(~G6;1)];
    bar b2;
    b2=[-b (~A6;1),(~A6;1),(~G6;1),(~;1)];
    bar b3;
    b3=[-b (~F6;1),(~F6;1),(~E6;1),(~E6;1)];
    bar b4;
    b4=[-b (~D6;1),(~D6;1),(~C6;1),(~;1)];
    /* declare a track variable attributes including instruments, fraction, beats per bar,
    beats per minute, and volume */
    track<<sax, 1,4,60,90>> t1;
    t1=[-t b1,b2,b3,b4]; /* assign value to the track variable with [-t …] with bars */
    track<<sax, 1,4,60,90>> t2;
    t2=[-t b3,b4];
    track<<sax,1,4,60,90>> t3;
    t3=toneUp(t2,2); /* use built-in function to raise the tone of track t2 by one degree*/
```

TUTORIAL sample

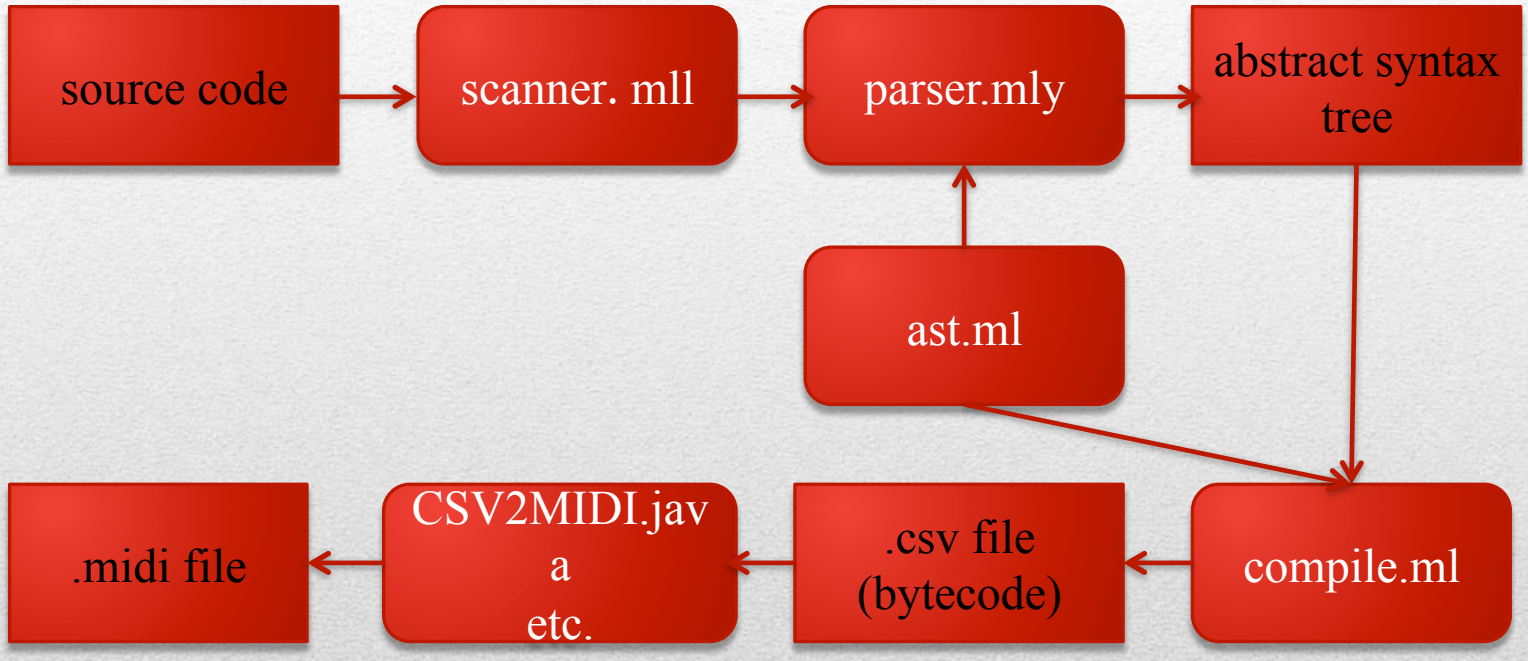
```
bar tmp;
/* use built-in function to get the first bar in track t3 and assign its value to bar tmp*/
tmp=at(t3,0);
note tmpn1; /* declare a note variable tmpn1*/
/* functions can be nested*/
tmpn1=toneDown(at(tmp,2),1);
bar newb;
/* assignment can be done with functions' return value*/
newb=[-b at(tmp,0),at(tmp,1),tmpn1,tmpn1];
t3=[-t newb,at(t3,1)];
track<<sax,1,4,60,90>> t4;
t4=t3+t3; /* tracks can be concatenated*/
track<<sax,1,4,60,90>> tall;
tall=t1+t4+t1;
track<<banjo,1,4,60,90>> tchor;
track<<>> tchor1; /* tracks would be given default attributes if not specified when declared*/
track<<>> tchor2;
track<<>> tchor3;
```

TUTORIAL sample

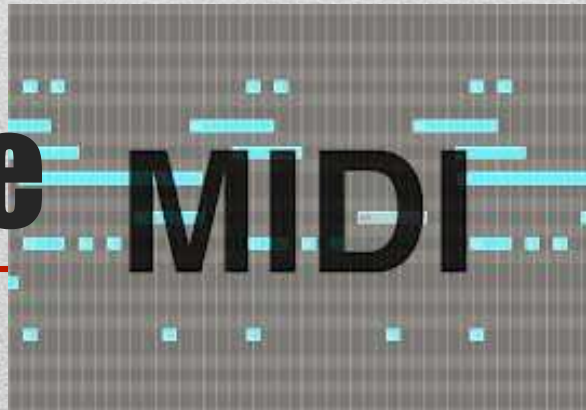
```
        tchor1=[-t [-b (~C4;1),(~C5;1),(~E5;1),(~C5;1),(~F5;1),(~C5;1),(~E5;1),(~C5;1)]];
        tchor2=[-t [-b (~D5;1),(~B4;1),(~C5;1),(~A4;1),(~F4;1),(~G4;1),(~C5;1)]];
        tchor3=[-t [-b (~E5;1),(~G4;1),(~D5;1),(~G4;1),(~C5;1),(~G5;1),(~B5;1),(~G5;1)]];
        tchor=tchor1+tchor2+tchor3+tchor3+tchor1+tchor2;

        melody twinkle; /* declare melody variable*/
        twinkle=syn(tall,tchor); /* synthesize tracks into one melody*/
        /* this returned melody value would be compiled and a .csv file would be generated*/
        return twinkle;
}
/*self-defined function to synthesize tracks into melody*/
function melody syn(track t1, track t2){
        melody m1;
        m1=t1&t2;
        return m1;
}
```

TUTORIAL sample

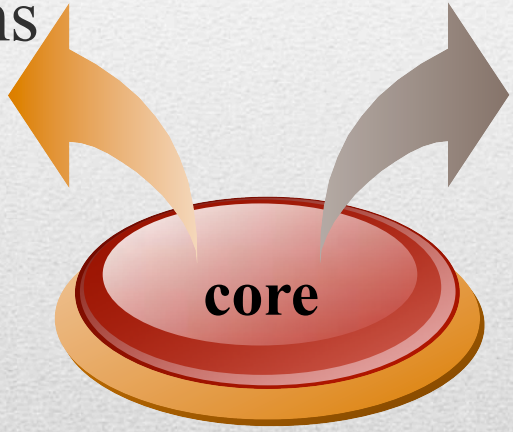


COMPILER_structure



- Define the data types of ourselves (element)
- Build functions

  mapstr2int()
  string_of_element()
  get_type()



- evaluation with semantic check
  - type check
  - input # of args check
  - ID validation check
- attributes extraction
- built-in func impl
- side-effect

Compile.ml

```
type element =
   Nte of int * int
  |Bar of (int * int) list
  |Tra of (int list) * ((int * int) list list)
  |Mel of (int list list) * ((int * int) list list list)
  |Rhy of int list
  |Pit of int
  |Lit of int
  |Stg of string
  |Bol of int
```

- -> Declare a variable and assign it at the same time
- -> Bar attributes extraction
- -> …

PROBLEM to SOLVE

. mc source file

([[105;4;2;1;1];[40;4;2;1;1]] ,

[[[(34,2);(250,2);(12,2);(12,2)];[(55,1);(78,1)]];

[[(34,2);(13,1);(88,2)];[(88,2);(81,2);(18,2);(22,2)]]])

. csv

4

105,40

0,34,90,0,34,1

4,20,90,1,13,1

5,12,90,5,88,1

7,12,90,7,88,1

9,55,90,9,81,1

13,78,90,11,18,1

,,,13,22,1

CODE GENERATION

. mc source file

([[105;4;2;1;1];[40;4;2;1;1]] ,
[[[(34,2);(250,2);(12,2);(12,2)];[(55,1);(78,1)]];
[[(34,2);(13,1);(88,2)];[(88,2);(81,2);(18,2);(22,2)]]])

. csv

4
105,40
0,34,90,0,34,1
4,20,90,1,13,1
5,12,90,5,88,1
7,12,90,7,88,1
9,55,90,9,81,1
13,78,90,11,18,1
,,,13,22,1

CODE GENERATION

CODE GENERATION

. mc source file

([[105;4;2;1;1];[40;4;2;1;1]] ,

[[[(34,2);(250,2);(12,2);(12,2)];[(55,1);(78,1)]];

[[(34,2);(13,1);(88,2)];[(88,2);(81,2);(18,2);(22,2)]]])

. csv

4

105,40

0,34,90,0,34,1

4,20,90,1,13,1

5,12,90,5,88,1

7,12,90,7,88,1

9,55,90,9,81,1

13,78,90,11,18,1

,,,13,22,1

LESSONS LEARNT

- Start early
- Work in big chunk of time
- Split the work into different chunks
- Group work to think of new idea/debug
- Ocaml is a good language for implementing compiler