OCaml Blockly

Kenichi Asai

Ochanomizu University, Japan

Thanks to Haruka Matsumoto and all the colleagues in my group
who contributed to the development of OCaml Blockly

October 13, 2025

Block-based programming environments

Scratch, (Google) Blockly, Alice, Snap!, etc.

- easy (or no) installation
- fun contents, such as animations, games
- community, where one can share created games

Technically, they provide

- language constructs as nested blocks (no syntax errors)

Slogan: Once a program is constructed, it runs.

Are you satisfied with the current status?

To run a program without an error, we have to deal with:

syntax errors a program must follow specified syntax

unbound values a variable has to be used within its scope

type errors a value can be used only as specified by the type system

For imperative languages, the latter two did not matter very much.

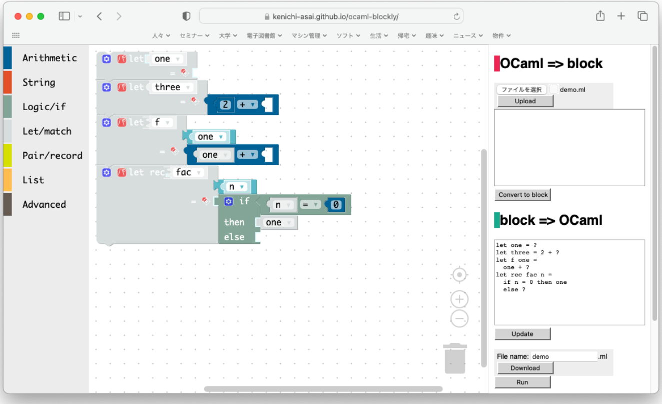
For typed functional languages, correct syntax is not enough.

Goal: to build an OCaml variant of Blockly (Demo)

where we encounter:

1. no syntax errors
2. no unbound values
3. no type errors

Once a program is constructed, it really runs.



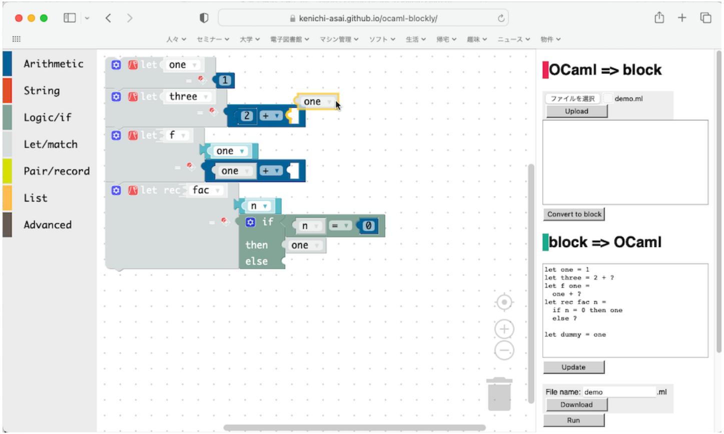
```
let one = ?
let three = 2 + ?
let f one =
  one + ?
let rec fac n =
  if n = 0 then one
  else ?
```

File name: demo .ml

No syntax errors

All blocks in menus

A variable is created by dragging it from a let block (with holes for arguments)



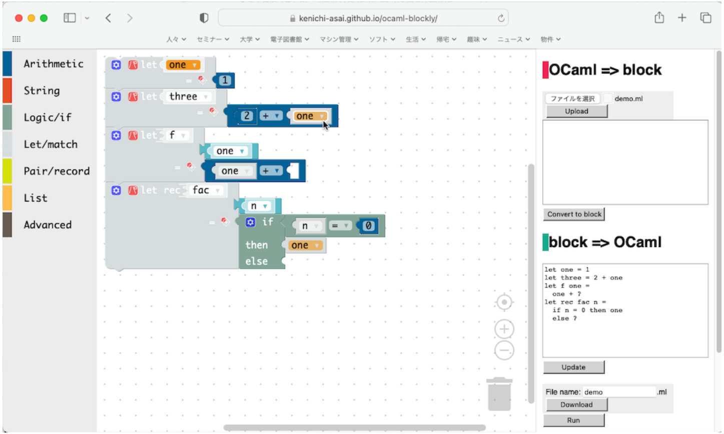
```
let one = 1
let three = 2 + ?
let f one =
  one + ?
let rec fac n =
  if n = 0 then one
  else ?

let dummy = one
```

No unbound values

A variable can be used only in its scope.

no variable capture

support α-renaming



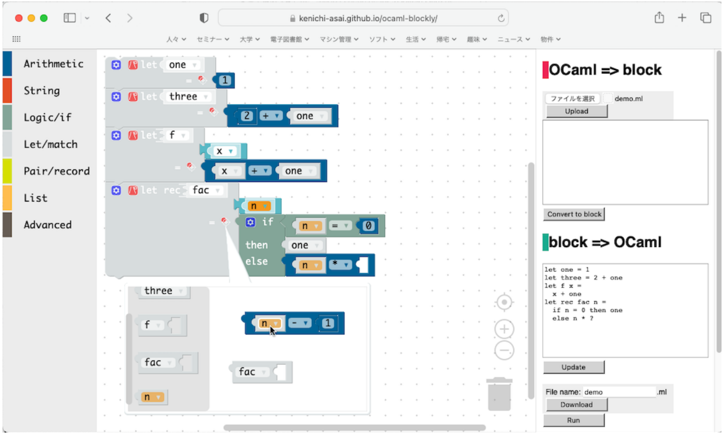
```
let one = 1
let three = 2 + one
let f one =
  one + ?
let rec fac n =
  if n = 0 then one
  else ?
```

Scope playground: bottom-up program construction

Variables cannot exist independently in the main playground.

Each let block comes with a scope playground.

- All the available variables in that scope are listed.
- They can exist on their own in the scope playground.



```
let one = 1
let three = 2 + one
let f x =
  x + one
let rec fac n =
  if n = 0 then one
  else n * ?
```

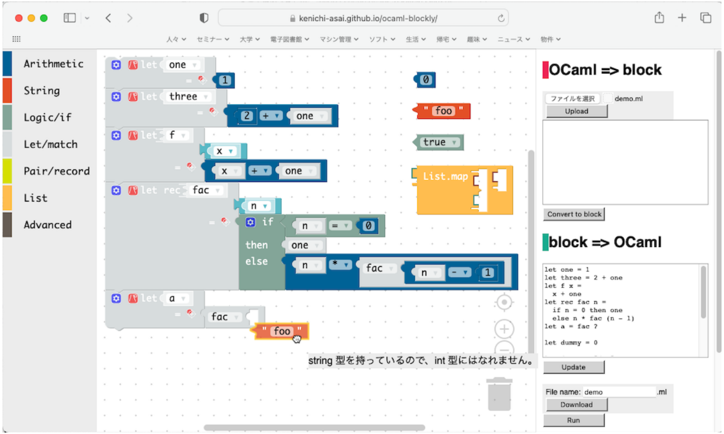
No type errors

For every user action, type checking is performed.

Connector shape shows a type.

Cannot connect blocks if its type does not match.

Supports let-polymorphism.



string 型を持っているので、int 型にはなれません。

```
let one = 1
let three = 2 + one
let f x =
  x + one
let rec fac n =
  if n = 0 then one
  else n * fac (n - 1)
let a = fac ?

let dummy = 0
```

OCaml Blockly

- No syntax errors, no unbound values, no type errors.
- Bottom-up program construction in scope playground.

In a nutshell,

OCaml Blockly behaves as we (language designers) expect.

So, how is it perceived by students?

OCaml course

- For 2nd year, CS-major students (around 40 students)
- 90 minutes flipped class, once a week, 15 weeks
- Design recipe
- Goal: a shortest-path problem for Tokyo metro network
- 2019, 2020 (in person), 2021, 2022 (online), 2023 (in person)

Students use either OCaml Blockly or Emacs. At week 10, they all have to move to Emacs, because OCaml Blockly does not support trees.

Research questions

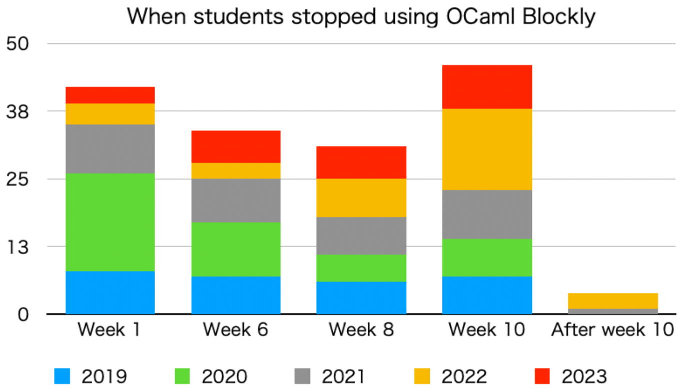
At the end of the course, I asked students:

1. When did students stop using OCaml Blockly?
2. Did students find OCaml Blockly useful?
3. Did programming become easier using OCaml Blockly?
4. Did OCaml Blockly have negative effects on text-based programming?

OCaml Blockly was not built as an educational research project.
It was build because I believed it would serve students well.

RQ1: When did students stop using OCaml Blockly?

- Some did not use OCaml Blockly at all
- Quite a number of students used OCaml Blockly as long as possible
- Some used it even after non-supported trees are introduced



Week 8: finish first shortest path problem
Week 10: trees

RQ1: When did students stop using OCaml Blockly?

From interview (2023) and free-text comments:

Week 1 Thought they can write programs in text. It's faster.

Week 6 Understood syntax.
Thought they need training in text-based programming.

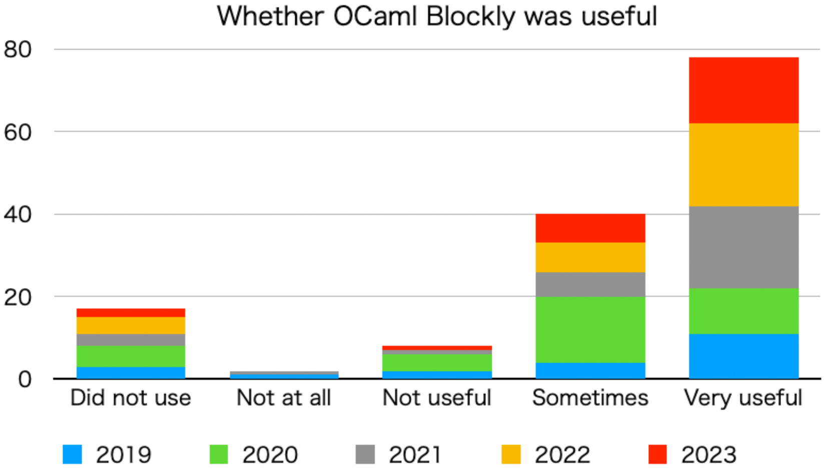
Week 8 Syntactic structure is clearer in block.
The length of programs does not matter.

Week 10 Postponed as much as possible since not remembered syntax.

After 10 For checking purpose.

Students acquire their own way of understanding programs in blocks.

RQ2: Did students find OCaml Blockly useful?



RQ2: Did students find OCaml Blockly useful?

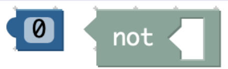
- Good only for small programs (−)

- At the beginning, writing programs in a text editor not realistic (+)
- Could see/understand structure of programs
- Concentrate on programming without being bothered by syntax
- Once used to OCaml Blockly, writing text programs became easier

- Useful for detecting type errors (on types)
- Could notice type mismatch when a block did not connect
- More aware of types even after moving to a text editor

Connector shapes and color

In OCaml Blockly, a type has its own connector shape.

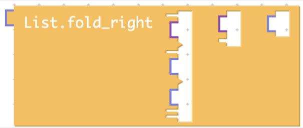


0

not

Does students remember connector shapes?

- Some do, some don’t.
- Remember simple types;
  do not parse complex types.

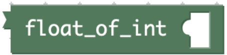


List.fold_right

Interviewee unanimously said they recognized colors.

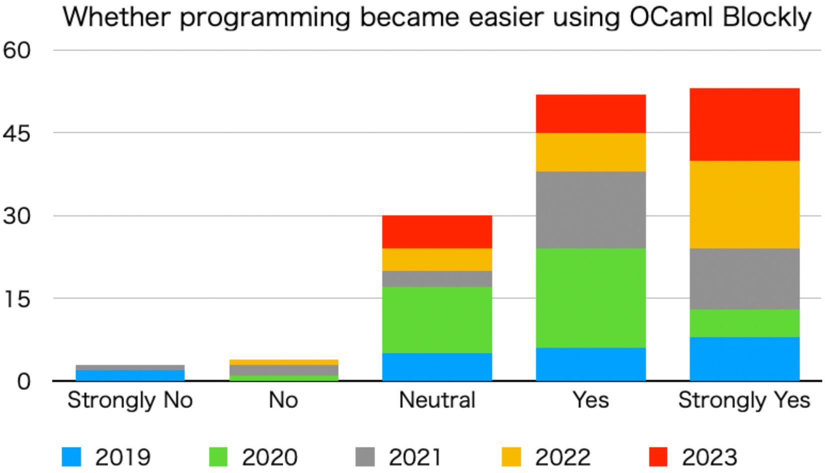
- Some even said she felt the ‘atmosphere’ of blocks.

Currently, colors of blocks in OCaml Blockly are based on types but not quite.



float_of_int

RQ3: Programming got easier using OCaml Blockly?



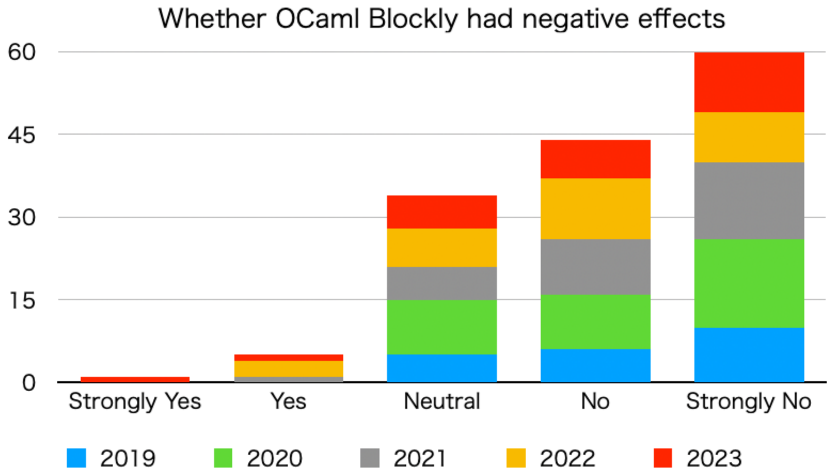
RQ3: Programming got easier using OCaml Blockly?

- (did not use OCaml Blockly) (−)
- Understood only when writing programs in a text editor

- Useful but not sure if it leads to deeper understanding (neutral)
- Too useful

- Syntax support useful (+)
- Typing support (more) useful. Able to think about types in general.
- Beginning students empowered. "Could think about programming positively," "I learned how to write a program from syntactic restriction," "Able to structure a program more easily."

RQ4: Did OCaml Blockly have negative effects?



RQ4: Did OCaml Blockly have negative effects?

- “I could get use to OCaml quickly via block interface.” (+)
- “Because of OCaml Blockly, I could catch up with the class.”
- “There were more benefits than drawbacks.”
- “I learned how to write programs in a text editor.”
- “Since it was the structure of program that was important, the interface did not matter.”
- “I became able to program in other programming languages, too.”

- Too useful; rely on it too much (neutral)

- Could not write any programs in text editor when switched (−)

Transition to text-based programming

Almost all: OCaml Blockly was nice at the beginning.

- They could get over syntax problem easily.
- "Without OCaml Blockly, it would have been difficult to write the shortest path problem."

- Confused when switching to text-based programming.

But:

- They became able to write programs at the end.
- Their ability to write textual programs would not be lower than the case they started it from the beginning.

Visual Studio Code with OCaml Platform?

VS code would be ideal for somewhat advanced students. It:

- signals syntax errors, type errors and warnings,
- suggests auto-completion,
- shows the type of a variable under cursor.

OCaml Blockly supports beginning students by showing

- what constructs are available in a menu,
- how complex constructs are built,
- how the use of a variable is restricted to its scope,
- how type checking is performed.

Summary and status

OCaml Blockly behaves as we (language designers) expect.

- Beginners do suffer from syntax/scope/type errors.
- OCaml Blockly supports beginning students.
- Transition to text-based programming does not appear to be hard.

Current status:

- After the main developer graduated, I maintain it (with difficulty).
- Trying to extract the essence (diffs) of OCaml Blockly with an undergraduate student, but she is graduating in several months.