> Remember

By Hugo Labrande
Issue #4 : Make your own retro text game with Moiki

We're changing up the format of this week's issue again, and here is a tutorial that'll show you an amazing new tool that will be useful for making your own text-based game for an 8-bit or 16-bit platform. Full disclosure: not a parser-based text adventure like we've been talking about in the last few issues; this is far too large a topic to fit in one issue of this newsletter! Instead, I will show you a very easy way to make branching games, the kind that ask the player to type a number to pick one of the choices offered. (Basically computerized "Choose your Own Adventure" books!) This is a very simple interface, and I'm sure a lot of you have tried to write a game like that when you discovered BASIC programming. But it's also very powerful: by building a complex structure and using variables, you'll be able to make exciting games ranging from computer novels to text-based RPGs.

This tutorial is very easy: **I especially recommend it if you don't know programming, or if you want to create a game for your retro computer with a young child. It should take about an hour**, all in all – by that I mean that in an hour, you'll have a new game that can be played on most retro computers!

The tool we'll be using this month, Moiki, is very streamlined, and its goal is to make creating games very accessible. It also exports code to Inform 6, which is the path (via the Z-Machine) to getting your game to run on a retro computer. Furthermore, if you know a bit of Inform 6, you'll be able to tweak Moiki's export to your liking, for instance if you need new functionalities, or a more powerful parser.

What is Moiki?

What is this amazing new tool? How come you have no idea what it is if it's that amazing? Well, the answer is: because it's French. ("Moiki" = "I'm the one who", as in "I'm the one who decides"!) Clément came on the French Interactive Fiction community's Discord server in 2020, with an idea for a tool which makes it even easier to author branching-narrative games. He put in a ton of work in his tool, and was very receptive to feedback; his goal was to make the experience as smooth as possible to entice new writers to start writing more text-based games. And it worked! We had quite a few newcomers that discovered our community thanks to Moiki, and a game written with this tool took second place in our annual competition. It's only very recently that Clément started translating it to English; in fact, the English translation of the website has only been live for a few days!

One area where Moiki shines is that Clément worked hard to ensure the story could be exported to a variety of formats: HTML, JSON, PDF (choose-your-own adventure book style), Ink (a major player in the branching narrative genre), and... Inform 6, Graham Nelson's language which can be compiled onto a Z-Machine game. The export is light enough that most games will fit in a z3 file (< 128kb), which means it'll be playable on most disk-based retro platforms, from the TRS-80 to the Game Boy,

the C64, the Beeb, and the Amiga. It could also run on tape: if it's smaller than 53,000 bytes, roughly, it can be run from tape on Commodore 64 (using the Ozmoo interpreter), just like Mini-Zork (and, now, Mini-Zork II). (However, I don't know if there is a ZX Spectrum interpreter for the Z-Machine that would run on tape; I know there is one for the Spectrum +3, but nobody has made one for tape, I believe.)

What you need

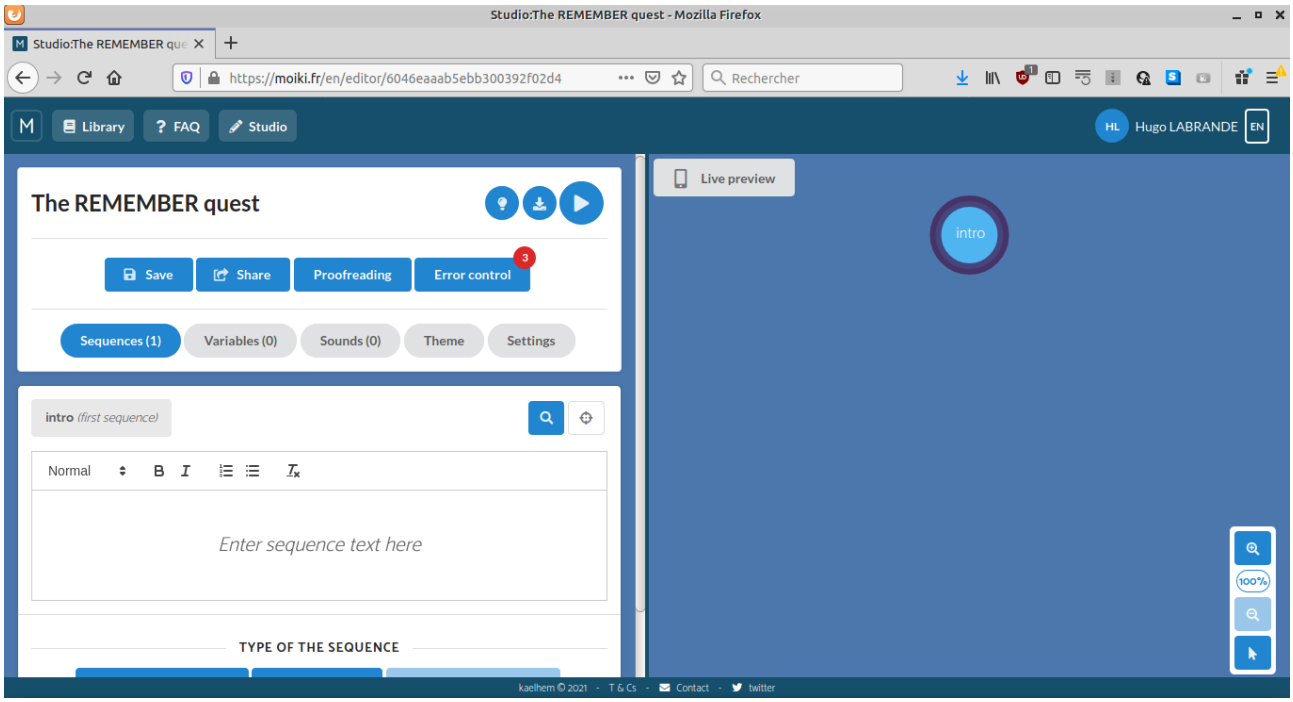
- Moiki.fr (it's probably a good idea to create an account)
- the latest version of the Inform 6 compiler, which right now is Inform 6.34, and can be found at the very bottom of this page
  http://www.ifarchive.org/indexes/if-archiveXinfocomXcompilersXinform6Xexecutables.html
- a Z-Machine interpreter for your favorite retro platform

Note that if your favorite retro platform does not have a Z-Machine interpreter, you could of course still write that kind of game another way, by writing it in BASIC for instance; but what I like about Moiki is that it's very visual and keeps your branching tidy, which really helps if your project starts to get big.

Making a text-based RPG

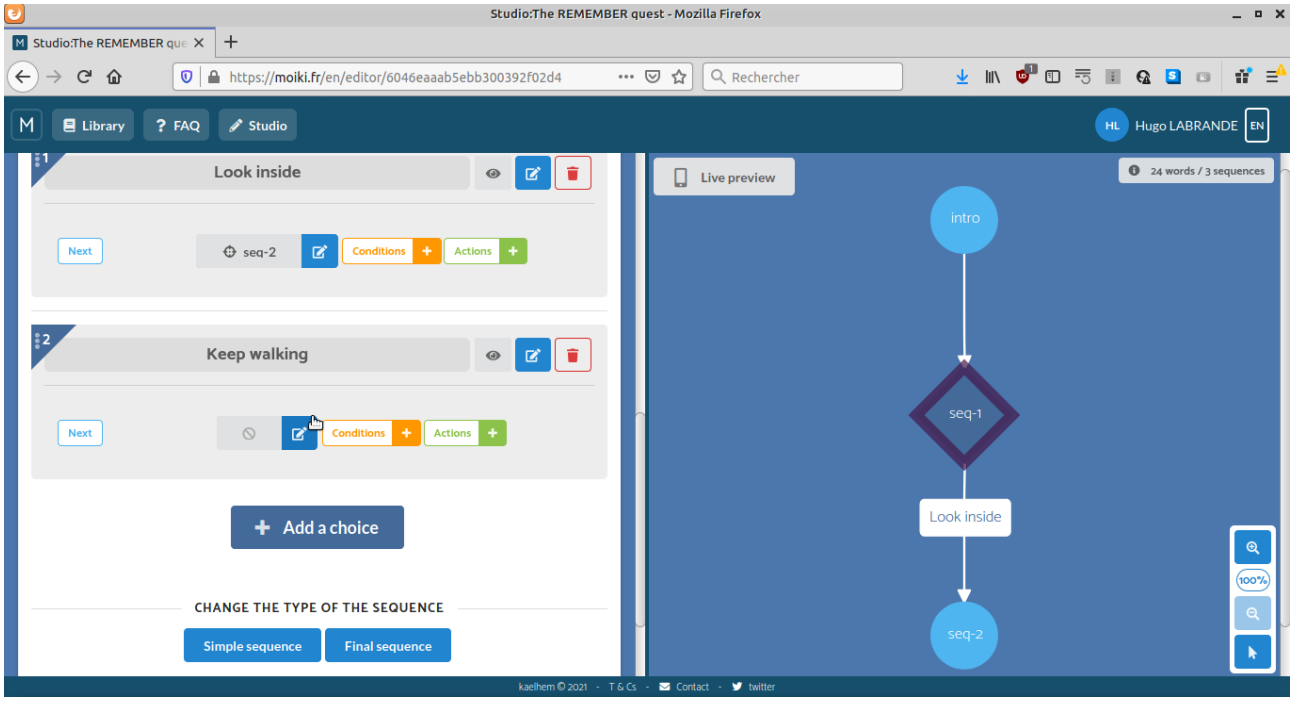
We'll be starting with something small: a couple of quests, changing statistics, fighting a monster. It should be enough to get your creative juices flowing! Start by going on Moiki.fr, and create a new story; give it a name, and a description, and let's go with "normal mode" – I'll explain everything.

First of all, in the top menu, you see a few buttons: "run" to test your story, "save" to save it on your account on the website, "share" to get a shareable link, "export" to export it to different formats; as well as a list of sequences and a list of variables.



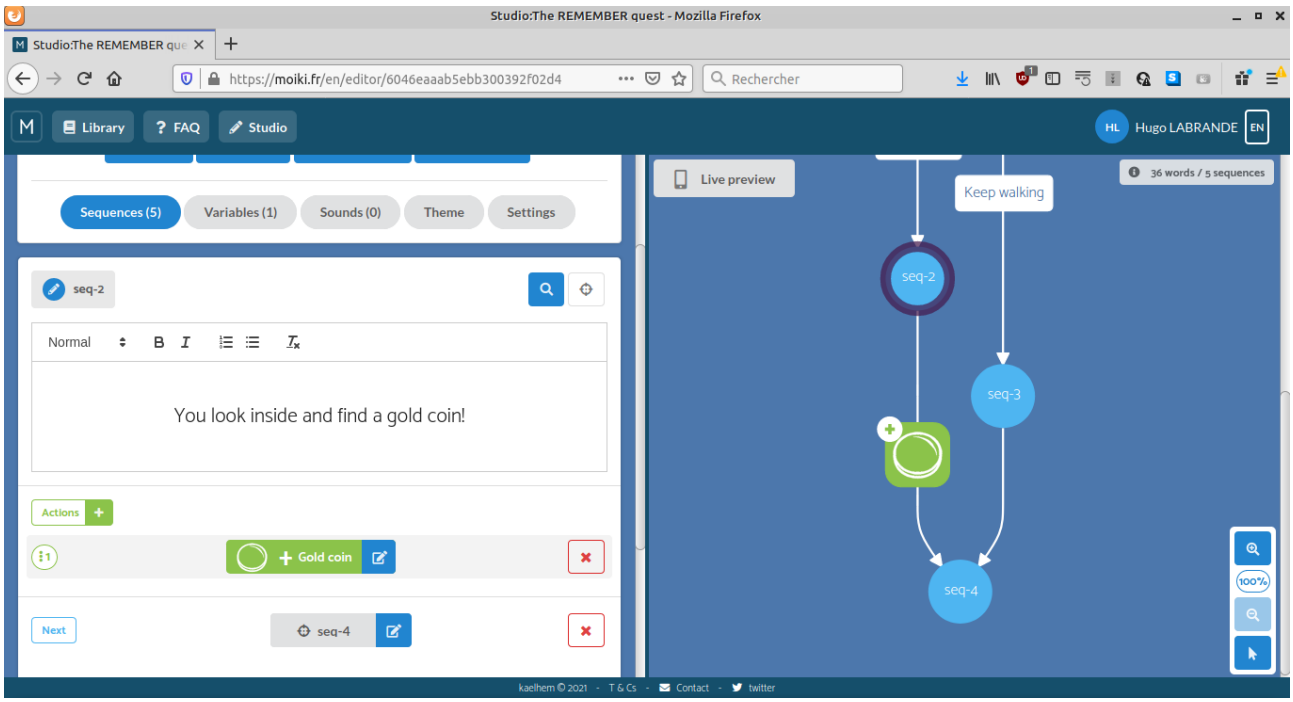
Your first sequence is already created: write some intro text, like “the king asked you to go forth and be a hero!”. Then, you can select the type of sequence it should be; let’s not do choices right away, so let’s choose to create a simple sequence. It should link somewhere, but as we don’t have another sequence yet, we need to click the “create a new sequence” button. A new node named “seq-1” appears, along with other options, but let’s not bother with these for now.

Click on the “seq-1” node to design the next step of your adventure. I’m going to go with the old well trope: do we want to look into it, or just keep walking? Write a little text corresponding to the situation, then select the type of the sequence to be one with choices. You are now prompted to type the first choice that should be given: let’s do “look inside the well”. A choice is created, and you see a “circle with a line across” icon below: this just means we haven’t said yet what sequence should the choice lead to. Let’s click on this, and the familiar popup appears; create a new sequence, seq-2. Now, add a new choice, “keep walking”, and create a new sequence, seq-3, as a result of that choice.

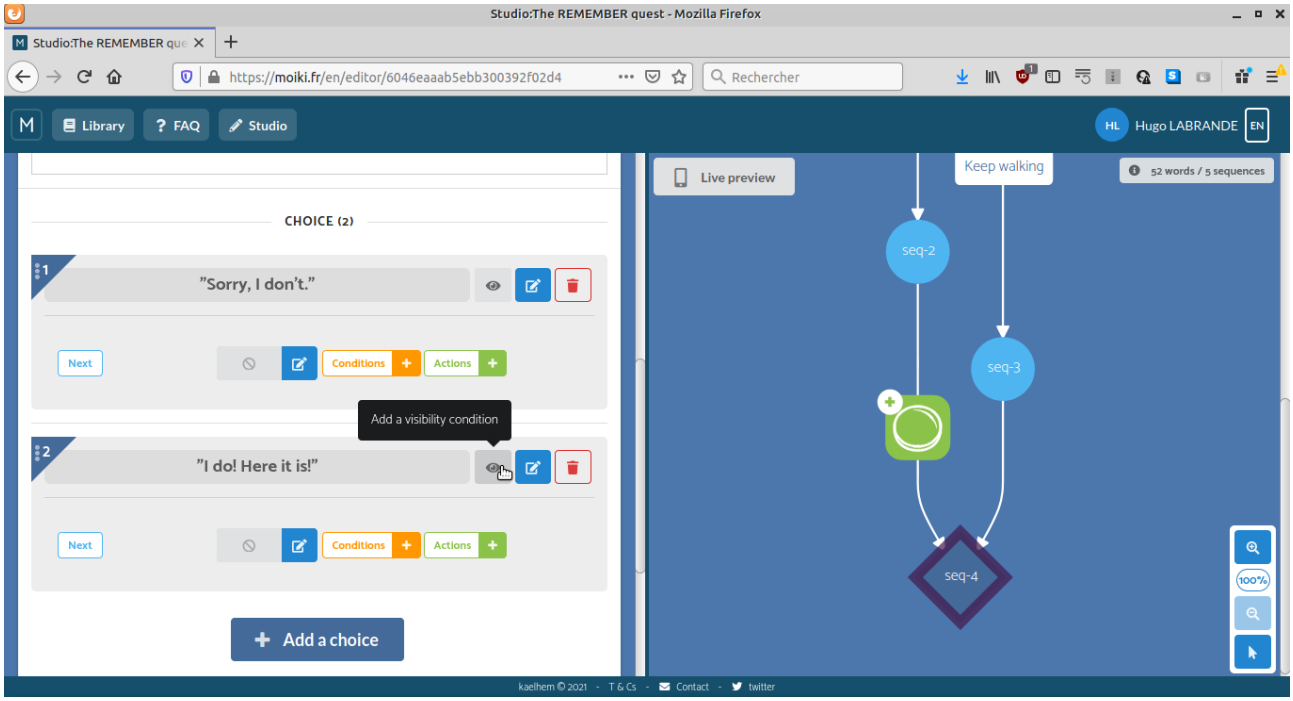


This simple decision about the well shouldn’t have too many consequences; at the end of the choices, we should probably go back to the main line of adventure. Let’s create it: go to “seq-3”, write something like “it probably doesn’t contain anything”, make that a straight sequence and create a new sequence, seq-4, that seq-3 leads to. You can now go to seq-2 and also make it a straight choice that goes to seq-4.

But wait! Shouldn’t we reward our player with something for looking into the well? How about a gold coin? First, create the gold coin: go to the “variables” button in the menu at the top, then in “object/hero” (currently empty) you should see a button to create one. Give it the name “gold coin”; you also need to give it an icon (which will appear on your graph in the right side of the screen, but also in the HTML – but not on our retro computer), by choosing one from the search engine. Now go back lower, to the seq-2 sequence (click on it in the graph to be taken to it). Below the sequence, you see an “action +” button; click it, then select “gain object” and select your gold coin.

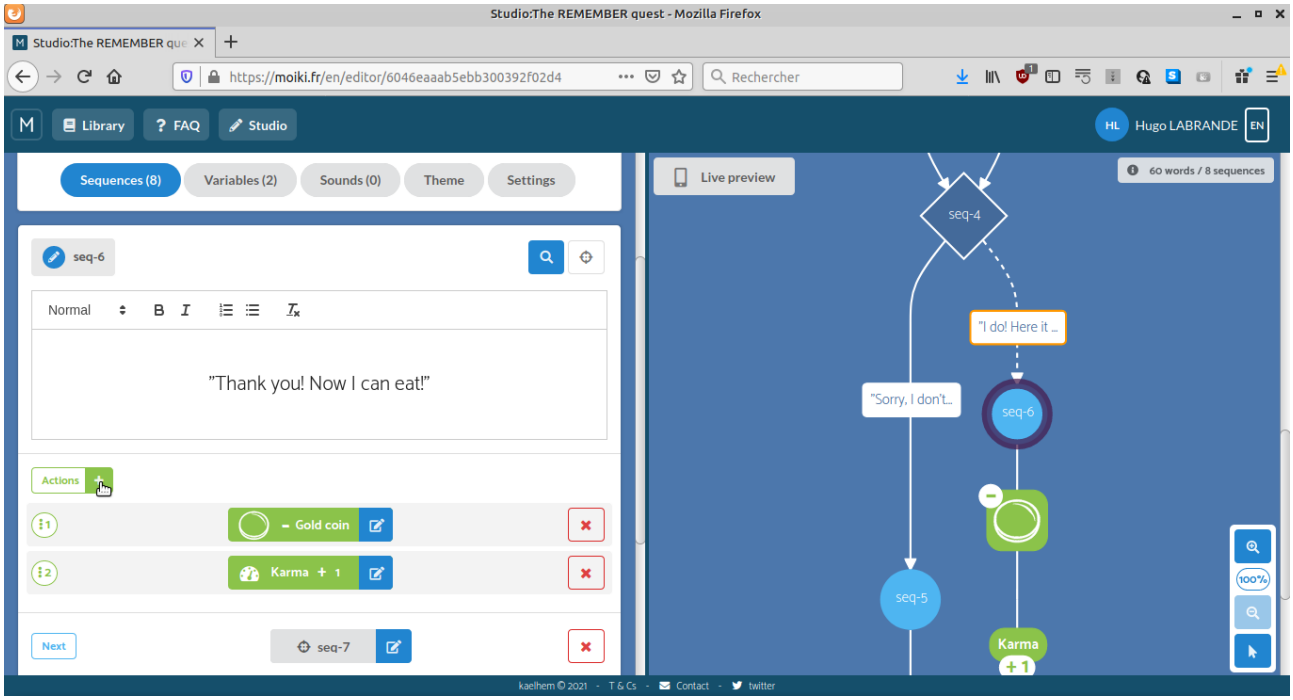


We have a gold coin? Let's have a beggar! We can give it a coin if we do, or apologize (or any other option you'd like). Again, select seq-4, write a text about the beggar, and add two choices, each leading to a new sequence (let's say seq-5 to give the coin and seq-6 to just apologize). But wait! We can't give a gold coin if we don't have one! So we need to control the visibility of the "give the coin" choice: look at the bar containing the name of the choice, and on the right side you should see an eye. Click on it, and you're invited to create a new condition: select "object", then "has the object...", and select the gold coin. The choice will now only appear if you have a gold coin in your inventory.

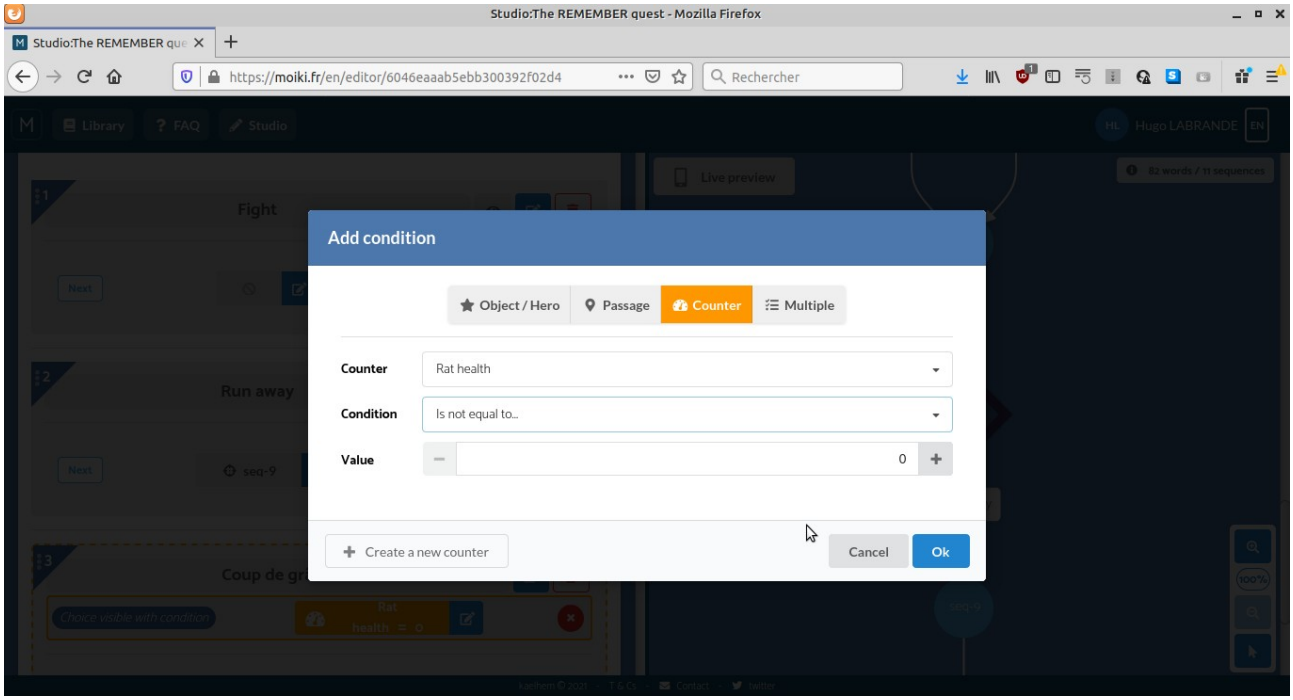


We should probably reward the player for giving a gold coin. Let's create a very simple karma system (or reputation, etc.); this is done by creating a counter, i.e. a

numerical variable. Go to the top menu, select “variables”; below your gold coin is a section on counters, which should be empty. Create a new variable, give it the name “karma”, ranging from 1 to 10 with a starting value of 5. Now go back to seq-5, the consequence of the “give the coin” choice; add an action, select “counter”, select “karma” in the dropdown menu, and say it should increase by 1. Finally, add another action, which is to lose the gold coin: “action +”, “object”, “lose an object”, “gold coin”. I’m sure you’re getting the hang of this already!

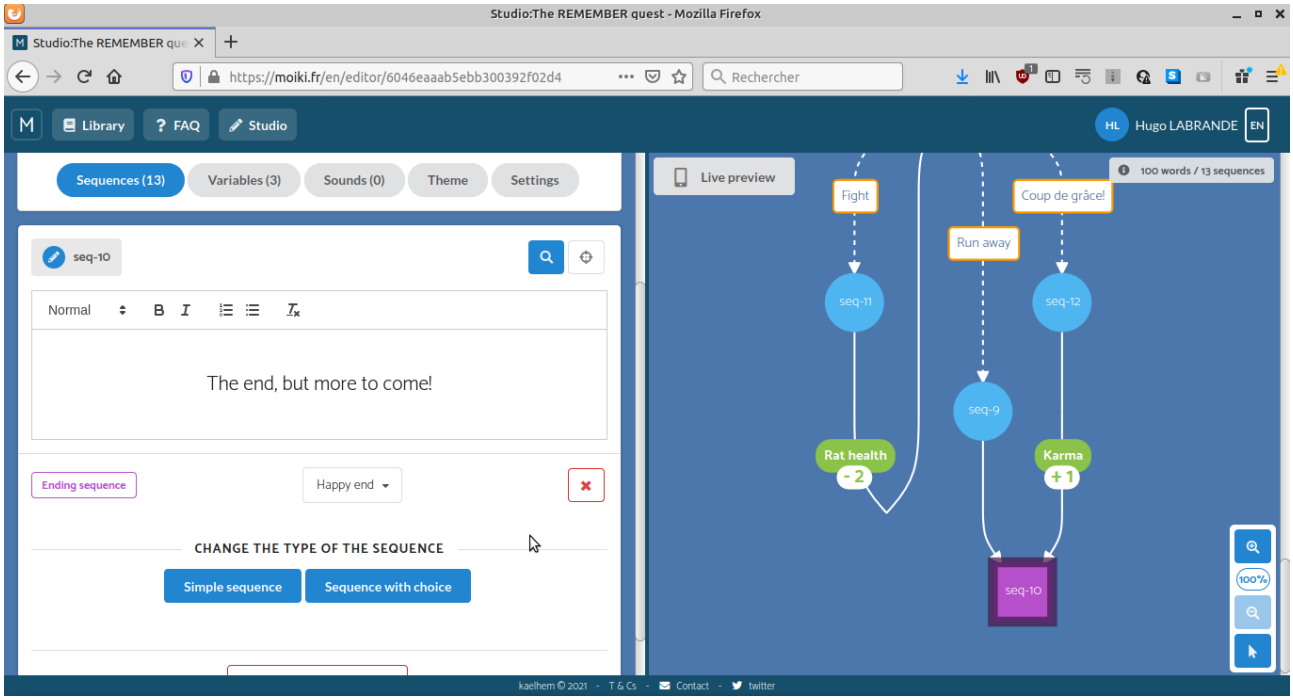


Don’t forget to tie seq-5 and seq-6 to a new sequence, let’s say seq-7. And now, let’s end this tutorial with a fight scene. I hear you sigh already: are we going to have to fight a rat with our bare hands? Yes, sorry... Let’s just create a very simple fight scene: your two choices are hit the rat, or run away. In seq-7 say “a rat is here! It attacks you!”; make this a simple passage leading to seq-8, the meat of the fight. Seq-8 should contain something like “The rat is ready to pounce!”, then the two choices: “fight” or “run away”. The “run away” choice leads to, say, seq-9, a simple sequence which says “you run away” and leads to the rest of the adventure (say, seq-10).



As for the fight sequence, let's have this very simple system: each attack takes 2 HP out of the rat, and it has 8 HP; when the rat's HP is at 0, the fight ends. Let's start by creating a new counter holding the rat's health: go up to variables, then create a new counter, starting at 8, max 8, and min 0. Then, inside seq-8, create a new choice, called "Coup de grâce!"; click on the eye icon on the right, and set this choice to only display when the rat's health is 0. As for the other two choices, click on their eye icon, and set them to only display when the rat's health is different from 0.

Now we just need to tie the loose ends: the "Fight!" should lead to a simple sequence, seq-11, saying something like "You hit the rat!", and leads back to our seq-8 node after decreasing the rat's health by 2; the "Coup de grâce!" choice should lead to a node, seq-12, in which you defeat the rat, maybe earn some karma, then go to the rest of the adventure (seq-10). Then, we can say that seq-10 is of type "final node", and write something saying "more to come!".



This is a very simple game, unlikely to capture anyone's attention for too long; but you probably did it in half an hour, no more. And you have 12 sequence, when Moiki can support literally hundreds. Imagine what you can do with this system with more time!

Tweaking the Inform 6 output

It's time to put our game on our computer. Click on the "Export" button in the top menu, and export your story as a JSON file. Save this file, then open the "Moiki exporter" app (in another window). Drop the ".zip" file you just got, and choose the "Inform 6" format. Tweak whichever settings you want (don't forget to specify the language as "en" to get the default messages in English!). You get a ".inf" file, and a ".h" file, which can just be edited with any text editor. The generated code is rather simple, and very lightweight, as the usual parser is sidestepped. If you don't know Inform 6, that's completely fine! Here's a couple of tricks you can do, though.

For now, Moiki doesn't support randomness, though it might come later. However, it is very simple to add it in your code. Look for the piece of code updating the rat's health; mine looks like

```
    subCounter(_rat_health_2, 2);
```

Here we could add randomness, to make this a little less predictable: let's say

```
    subCounter(_rat_health_2, random(4));
```

Now each hit will deal damage between 0 and 3!

Another way to make the game better is to change the line "You hit the rat!" so it's not always the same. You can replace

```
    print "You hit the rat!"
```

with

```
    print random("You hit the rat!", "You kick it, but it tries to fight back!", "You step on its toes!");
```

This makes your game change the line randomly, which is good for lines you see all the time.

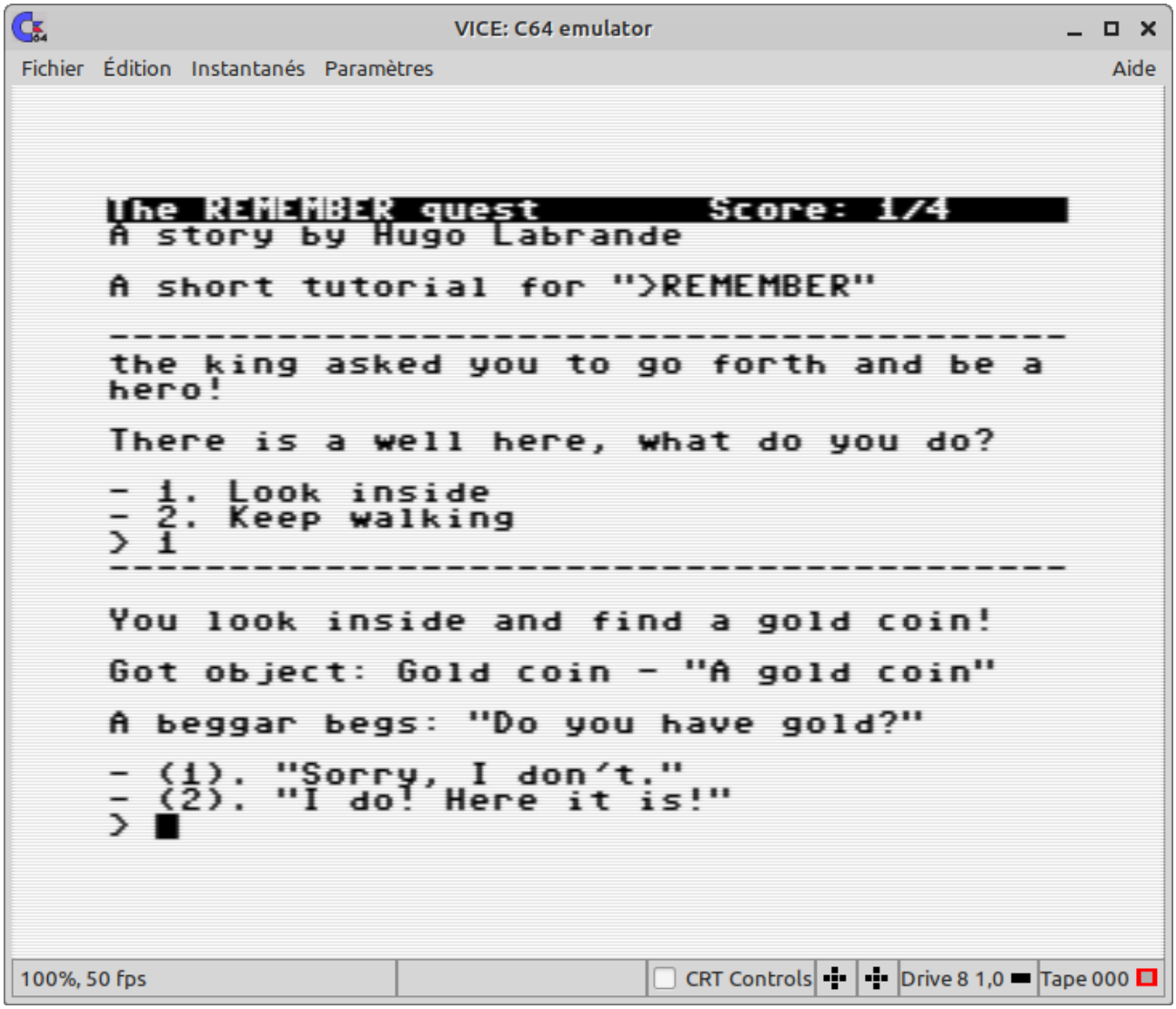
Careful, though! You're tweaking the output of Moiki; if you decide to go further on Moiki and then export the result, those changes will be lost! It's probably better to save them for last, or copy-and-paste them in periodically. I'll let you figure out how you want to organize this.

Once you're done, you should compile your game. This is done through the command line (cmd.exe on Windows, a terminal on OSX or Unix); in theory, just typing

```
inform -v3 story.inf
```

should work. The resulting file is a "game.z3" file that only weighs a few kilobytes (mine is 4.5kb) – you can put it on a disk with a Z-Machine interpreter and run it! And there you go, you've authored a text RPG for retro systems!

The quick and easy way to generate a C64 game is to use Ozmoo Online:
http://microheaven.com/ozmooonline/



What I particularly like about Moiki's export is how small it is, and it gets compiled to efficient Z-Machine code. Most Z-Machine interpreters on 8-bit machines only have z3 support, which means you cannot go further than 128kb, although there are some 8-bit computers that should support z5 (like the Apple II, the C64, and the BBC Micro); once you're on 16-bit machines, you probably can find a z5 (256kb) or z8

(512kb) intepreter. Plus, thanks to the Z-Machine's text encoding scheme (see Issue #2!), you can fit around 25% more text; and if you want to use abbreviations, they probably would free up more than 20k on a file that's getting too close to the 128k limit! My back-of-the-envelope calculation is that a Moiki-generated game with up to 200,000 characters of text could fit in a z3 file – that's 40,000 words of adventuring, which I'm fairly certain is the size of a book of the "Choose Your Own Adventure" line! If your 10-year-old self was wondering if a computerized adaptation of "Deathtrap Dungeon" would ever run on the old Acorn Electron, it might very well be possible...

I hope you have enjoyed this month's article! I think the Moiki tool has a ton of potential to introduce people (you, your children, your grandchildren, and more!) to writing branching narratives, text RPGs, computer novels, and all of that, and I'd be delighted to hear about your creations. Feel free to reach out to me if you have any comments on the tool, I can pass it on to its creator if needed! I'll see you next month!