The design of

Splash!

A NEW KIND OF SYSTEM DYNAMICS SOFTWARE

CREATIVE LEARNING Exchange
Making Thinking Visible

BTN

The design of Splash! - a new kind of system dynamics software by The Creative Learning Exchange and BTN Pte. Ltd.

The Creative Learning Exchange

27 Central St.
Acton, MA 01720
USA
http://www.clexchange.org/

BTN Pte. Ltd.

28 Bukit Pasoh Road
Yee Lan Court
Singapore (089842)
http://learnwithbtn.com/

First published online on 5th November 2016.
Get the latest version of this document online at www.clexchange.org/splash
or www.learnwithbtn.com/splash

Acknowledgements

The Design Team

Anne LaVigne
David Wheat
Diana Fisher
George Richardson
Lees Stuntz
Ninad Jagdish
Tracy Benson
Warren Farr

Contents

1. What is Splash?

Liquid Physics + System Dynamics = Splash!

Whether it's managing your health, planning your retirement or addressing climate change, System Dynamics[1] (SD) can help us tackle the wide variety of dynamic problems we find in the world around us.

There are several software available in the market to create system dynamics models. These tools are often highly capable and flexible products that are designed to help advanced users. But while these software may be very suitable for professionals, they can end up being intimidating and boring for those who are just starting to learn system dynamics.

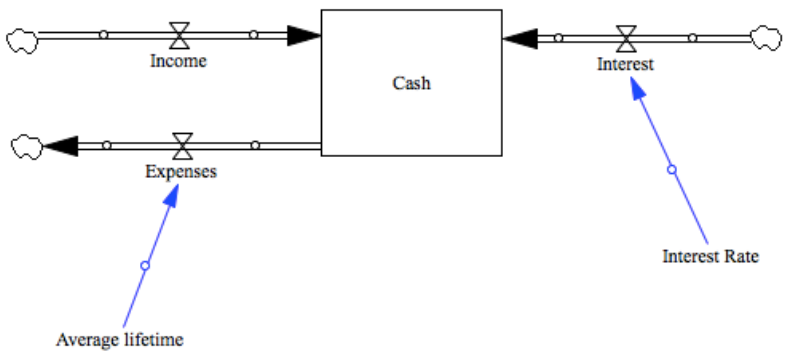
There is an evident need for a simple modeling software that's designed with beginners in mind – something that middle and high school students might use to learn system dynamics. If this tool were to exist, it would make system dynamics accessible to a much broader audience and help catalyze its use.

Splash is our attempt at dreaming up exactly such a software. Designed primarily for tablets and mobile devices, Splash combines liquid physics simulations with system dynamics in a way that emphasizes fun, delight, and ease-of-use as much as it does the core principles of system dynamics.

2. What will a Splash model look like?

Consider the simple system dynamics model of a bank account as shown in figure 2.1. This image is meant to be representative of what the model looks like in existing system dynamics software.

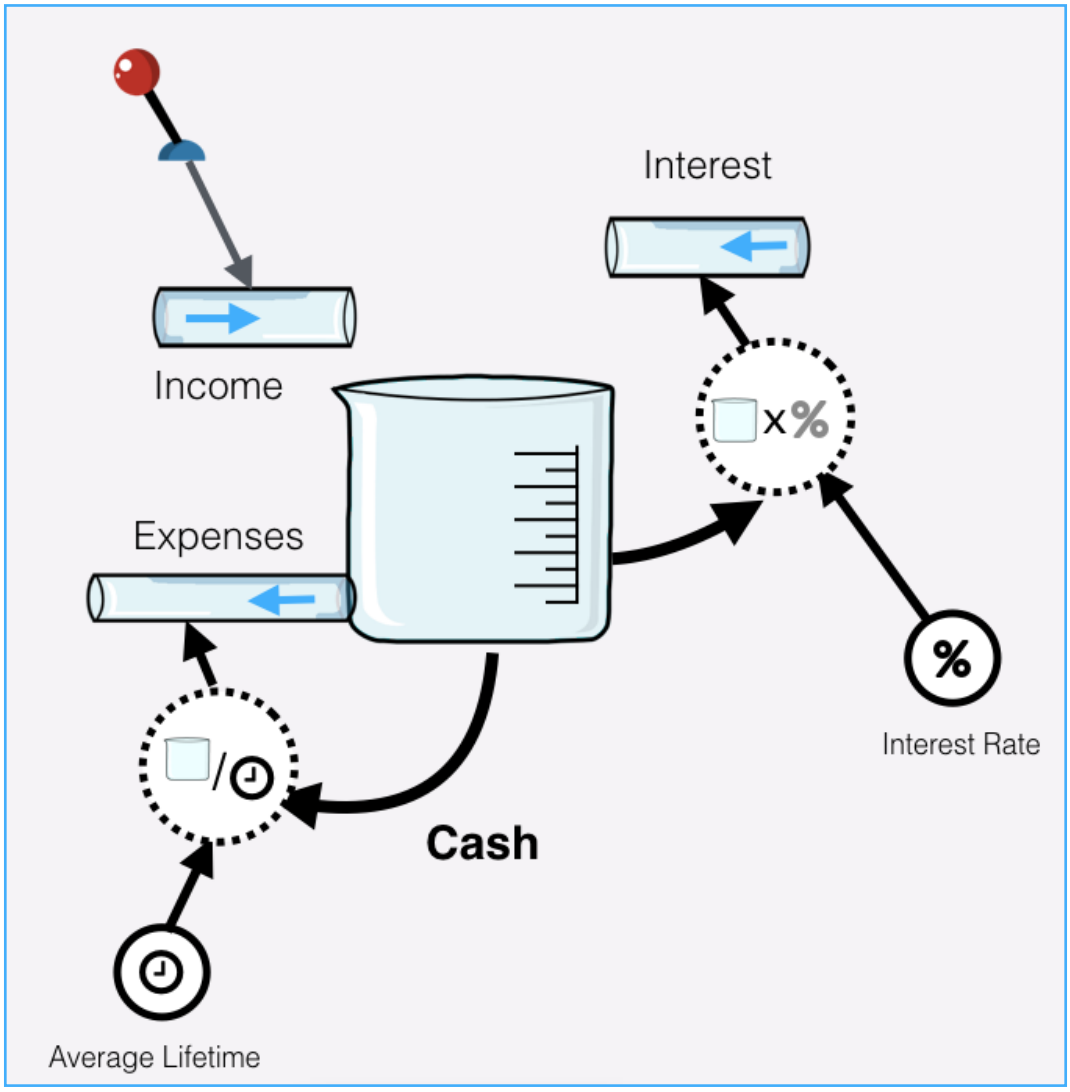
Fig. 2.1: A bank account model in existing SD software



[1] The term System Dynamics here is used to refer to both systems modeling and systems thinking.

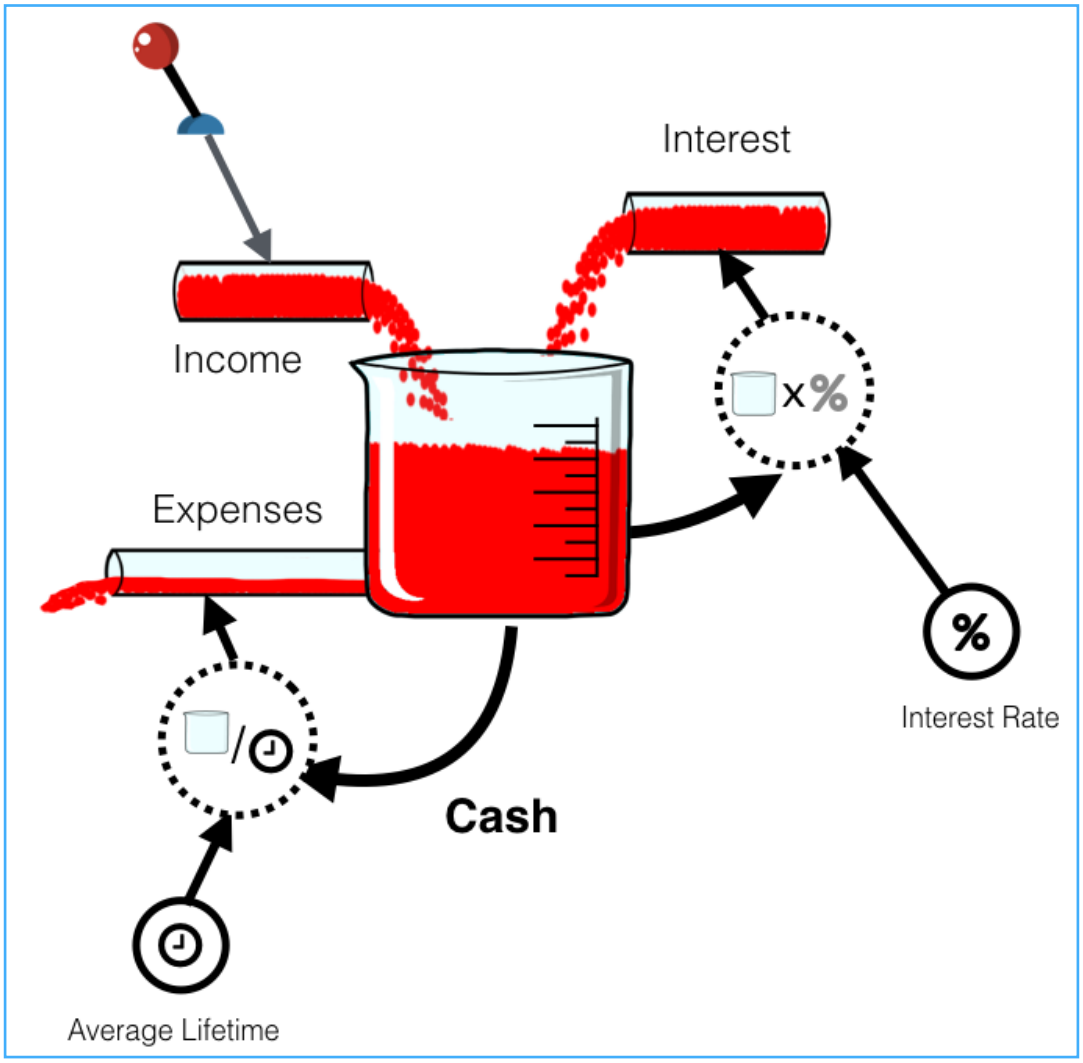
And here's what the same model will look like in Splash:

Fig. 2.2: The same bank account model in Splash



In Splash, stocks will be represented by physical *containers*, flows will be represented by *pipes* and auxiliary variables will be represented by circles. In addition to these familiar kinds of system dynamics elements, Splash will also have visual *mathematical operators* – elements that can be added into the model to specify the equations. Want to multiply two variables together? Drag the multiplication operator and connect the two variables to it. Want to divide two variables? Use the division operator. By introducing operator elements, Splash will make math more visual and make models more transparent. A variety of *control elements* will also be available for use in Splash. These include levers, sliders, switches, step inputs and pulse inputs. When you're making a model, you'll be able to connect these control elements to flows and auxiliaries and conveniently change their values while the model is being simulated.

Fig. 2.3: Simulating the bank account model in Splash



Speaking of simulating the model, take a look at Figure 2.3. When you simulate the bank account model in Splash, a virtual liquid representing money will flow through the Income and Interest pipes and fall under gravity into the Cash container. Money will flow out of the Cash container through the Expenses pipe and disappear or simply fall off the screen. The amount of stuff going through the inflow and outflow pipes will be determined by the equations or control elements that drive them. For example, the amount of stuff flowing through the Interest pipe will be determined by how much stuff is in the Cash container and what the Interest Rate is.

What is of importance is the fact that this visual appearance of matter flowing through pipes and accumulating in containers is not a mere animation. It will be based on simulations of physics. In order to do this, Splash will have a physics engine built right into it. This will allow users a new dimension to experiment with. Splash will create a flexible playground where users can come up with innovative arrangements to represent SD models. For example, consider the model shown in Figure 2.4.

Fig. 2.4: CO2 model in existing SD software

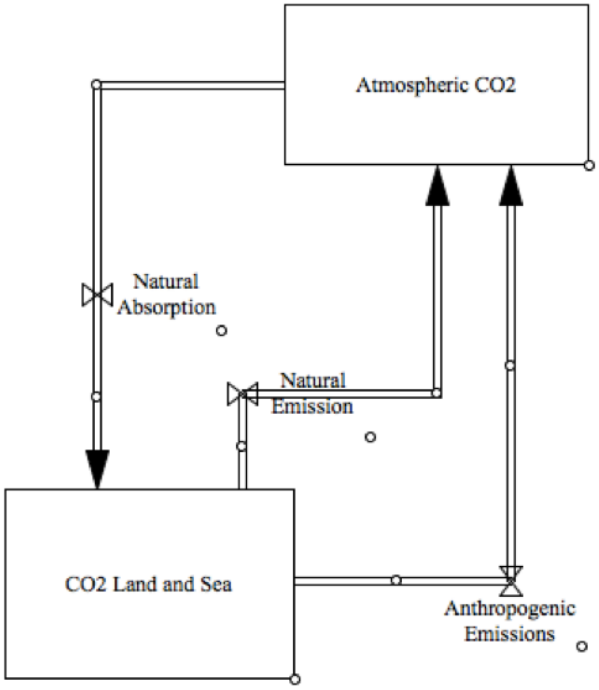
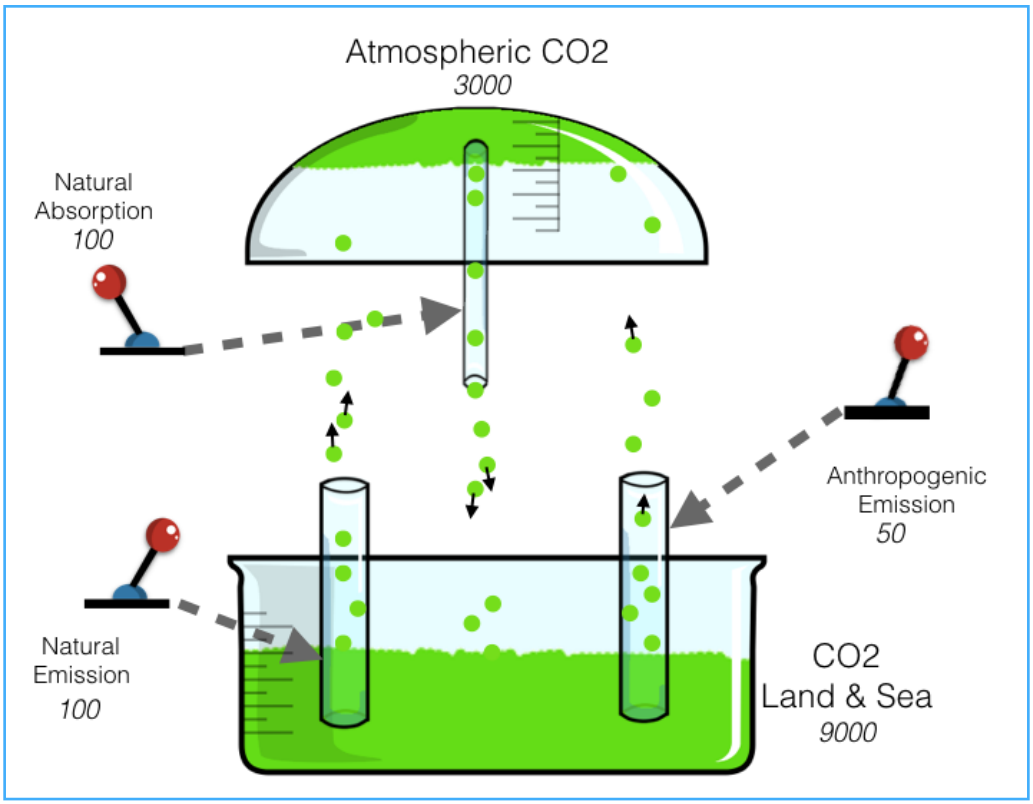


Fig. 2.5: The CO2 model in Splash



It's a simple model on CO2 emission and absorption. There's a stock of CO2 in the land and sea and a stock of it in the atmosphere. Natural and Anthropogenic Emissions send CO2 from the Land and Sea stock into the atmosphere. CO2 in the atmosphere gets absorbed into the Land and Sea stock over time and this is captured by the Natural Absorption flow.

Figure 2.5 shows how the same CO2 model is represented in Splash. The Land and Sea CO2 stock will be built like a trough at the bottom, while the Atmospheric CO2 stock will be created using an inverted bowl-shaped container at the top. All the flows will be set to invert the gravitational force on the liquid flowing through them. The emission flows will take the CO2 from the Land and Sea stock and pump it up into the atmosphere, while the Natural Absorption flow will make the CO2 from the atmosphere fall back into the Land and Sea trough. In this way, the model in Splash may not only be more fun to watch but will also mimic the spatial nature of the actual CO2 cycle.

With these examples, you hopefully get a sense of what a model in Splash will look like and how the app might be different from the existing system dynamics modeling tools. But these images only give an overview of Splash's concept. We've put a fair amount of thought into the details of how Splash will actually work. We've designed its information architecture, user interface (UI) and the entire user experience (UX). If you're interested, you can find more information about our design process and how the design evolved in ***Behind the Scenes***. For now, read on to see how you might use Splash to do all the things you normally do with an SD software and then some.

3. How will I use Splash to…

3.1. Build a system dynamics model

When you open up Splash, you'll see a Model Library (figure 3.1.1) listing all the models that are on your device. The Model Library is where you will get to access and manage your models. You'll be able to search and sort through them, rename, duplicate or delete them and even share them with other people. Tapping on the '+' in the Model Library grid will create a blank *Creation Canvas* on which you will build a new model (see figure 3.1.2).

On the sides of the Creation Canvas, you'll have two toolbars. The toolbar on the left will have containers (for stocks), pipes (for flows), control elements, auxiliaries and connectors. The toolbar on the right will have all the basic arithmetic operators you might need to create equations. It'll also have a special tool for creating graphical functions. Tapping on a tool will activate it. Once the tool is activated you'll tap on the screen to add the corresponding element into the model.

Fig. 3.1.1: The Model Library

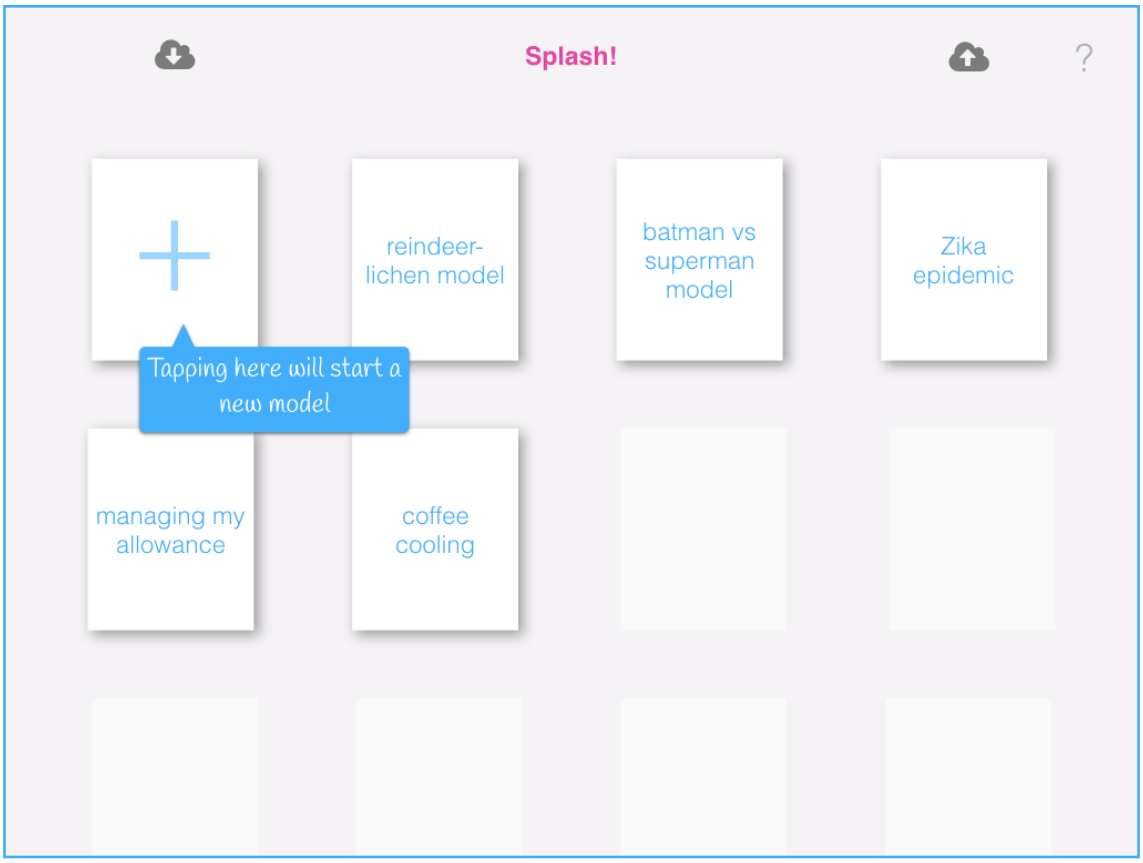


Fig. 3.1.2: A blank Creation Canvas

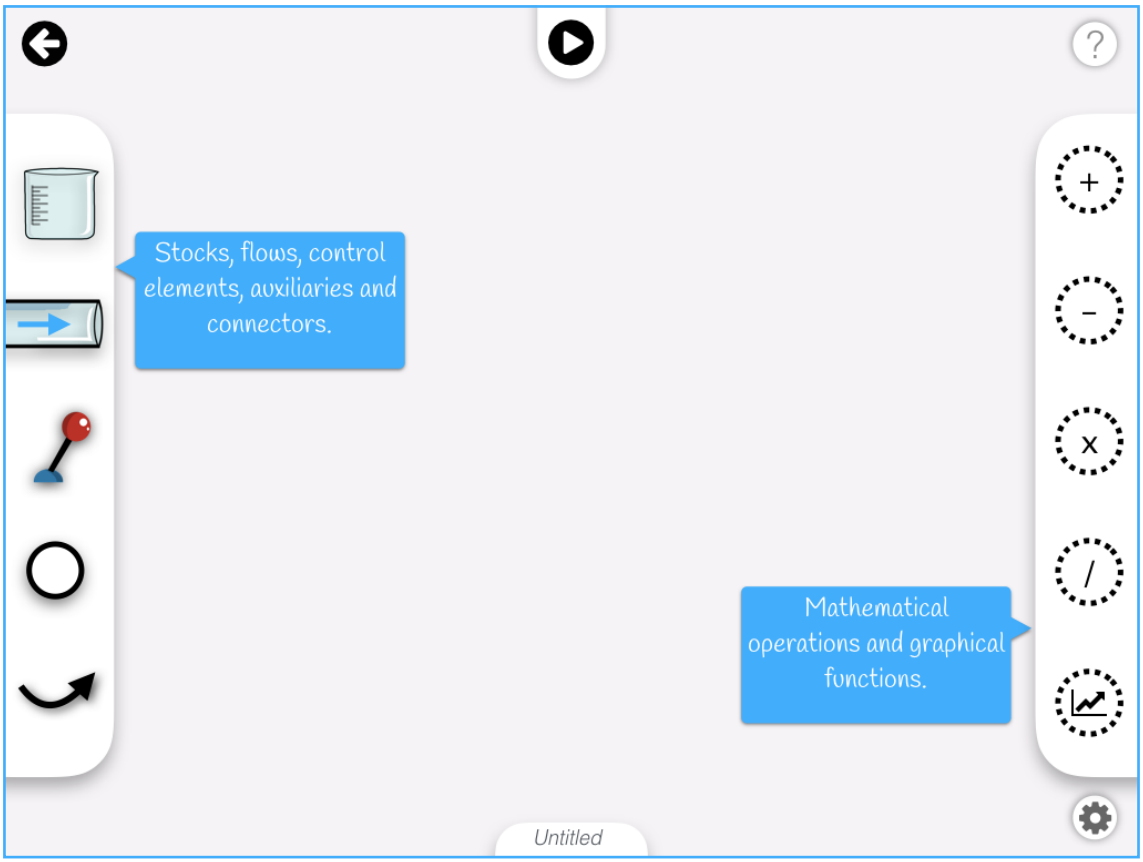
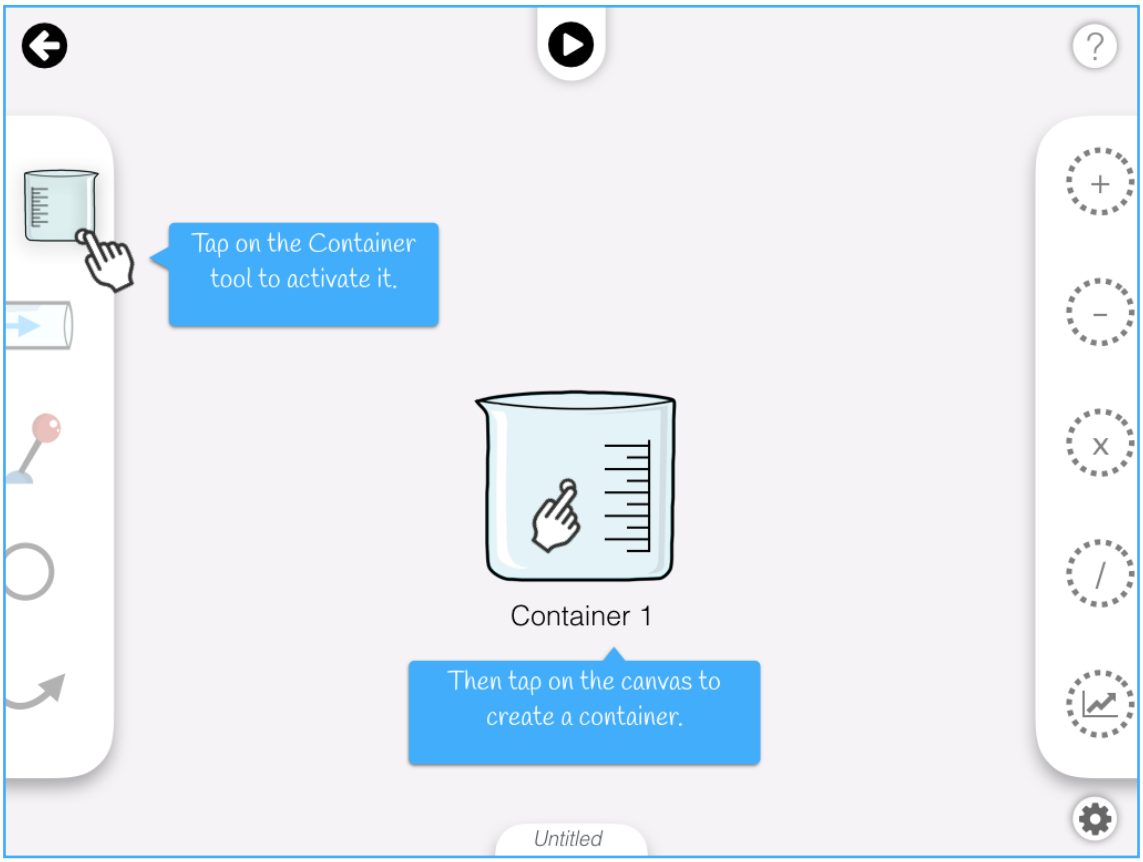


Fig. 3.1.3: Adding a container for Cash



For example, to create the Cash container you saw in the bank account model, you'll first tap on the Container icon in the left toolbar to activate it and then tap on where you want the stock created on the canvas (refer Figure 3.1.3).

To create the Income pipe, you'll tap on the Pipe icon in the toolbar and then tap and drag on the canvas to create the flow in the direction that you want it (Figure 3.1.4). Auxiliaries, control elements and connectors would also be added into the model in a similar way. If you want to reposition any of the objects you create, you'll simply have to drag them over to where you want them to be (Figure 3.1.5). The same applies to repositioning the object titles. Tapping on a title will bring up a keyboard with text formatting options for you to edit it. Figure 3.1.6 shows what renaming 'Container 1' to 'Cash' would look like.

Auxiliary variables in Splash will be represented by circles (Figure 3.1.7). To help distinguish between multiple variables, you'll be able to add a symbol for each auxiliary from a set of pre-built repository of icons. To do this, you'll tap on the auxiliary to bring out its *Property Panel.* Via the Panel, you can select a symbol, change the auxiliary's color, re-name it and adjust its default equation value (Figure 3.1.8). Containers, pipes, connectors and control elements will also have similar Property Panels.

Fig. 3.1.4: Adding flows to the model

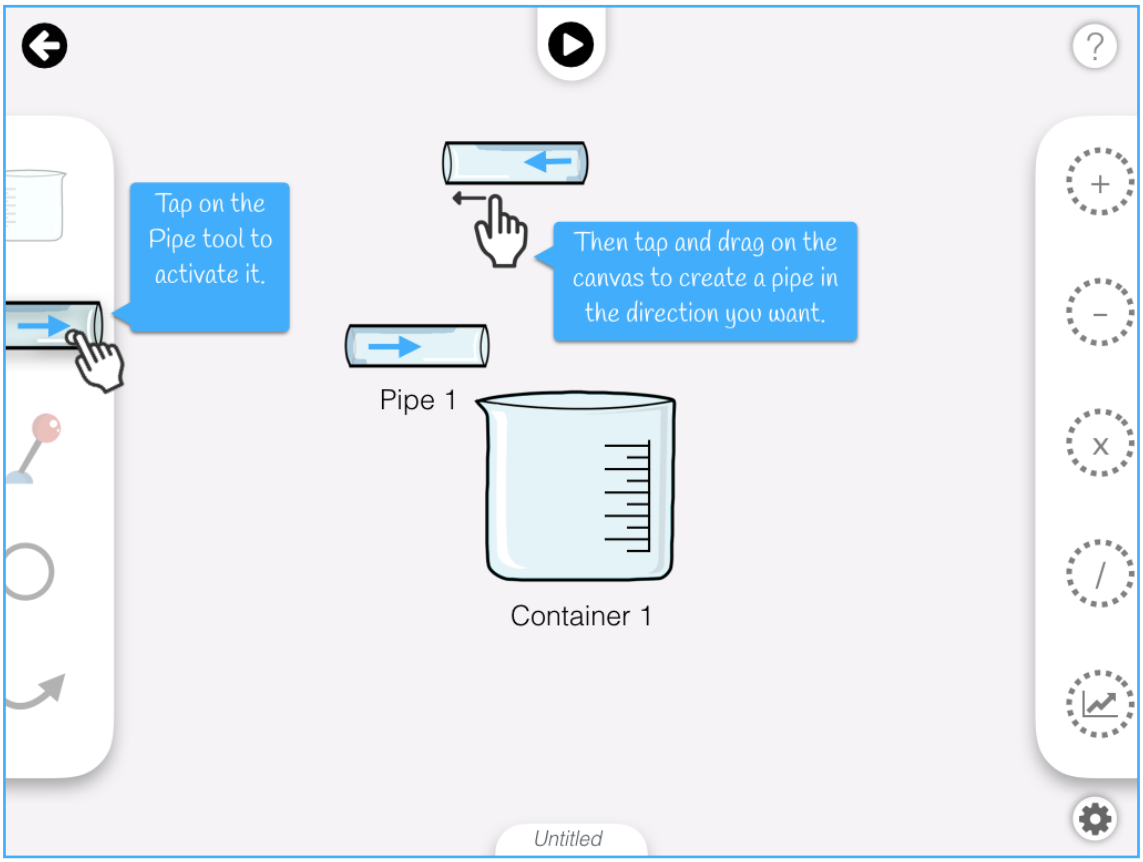


Fig. 3.1.5: Repositioning objects

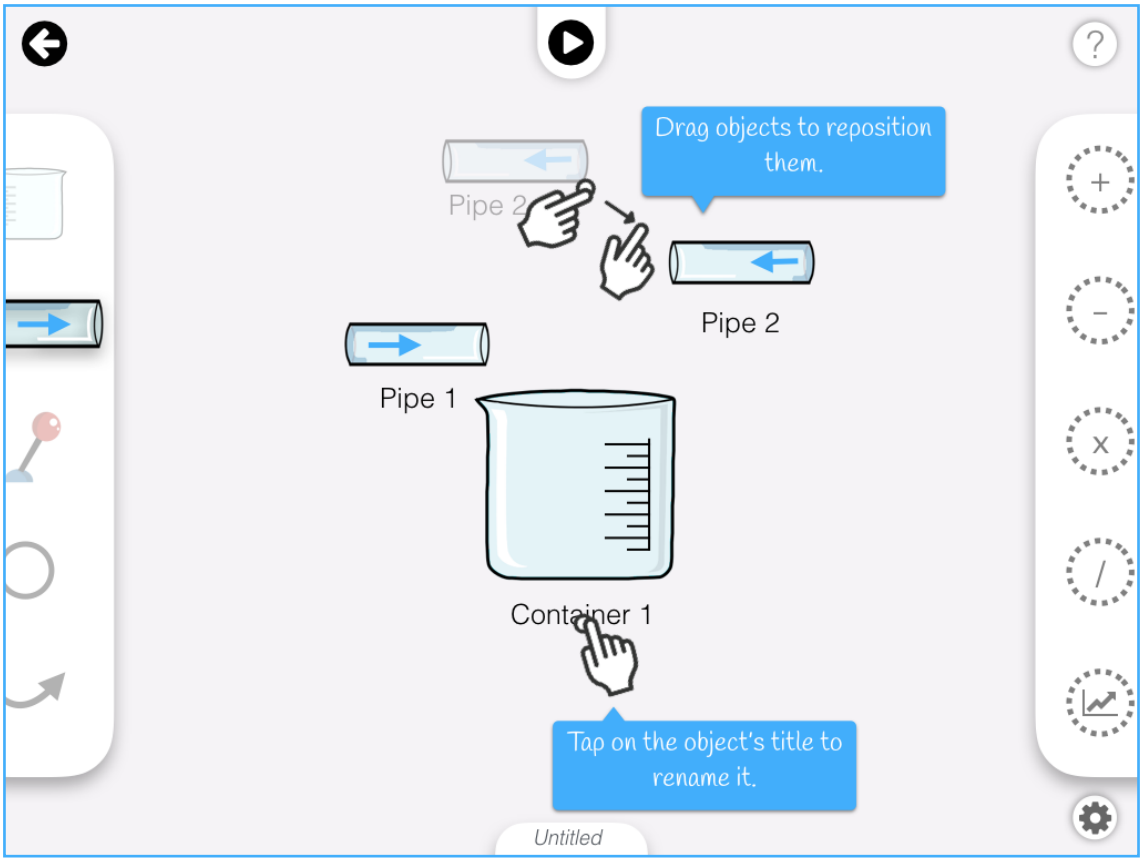


Fig. 3.1.6: Naming the Cash container

Fig. 3.1.7: Auxiliaries represented by circles

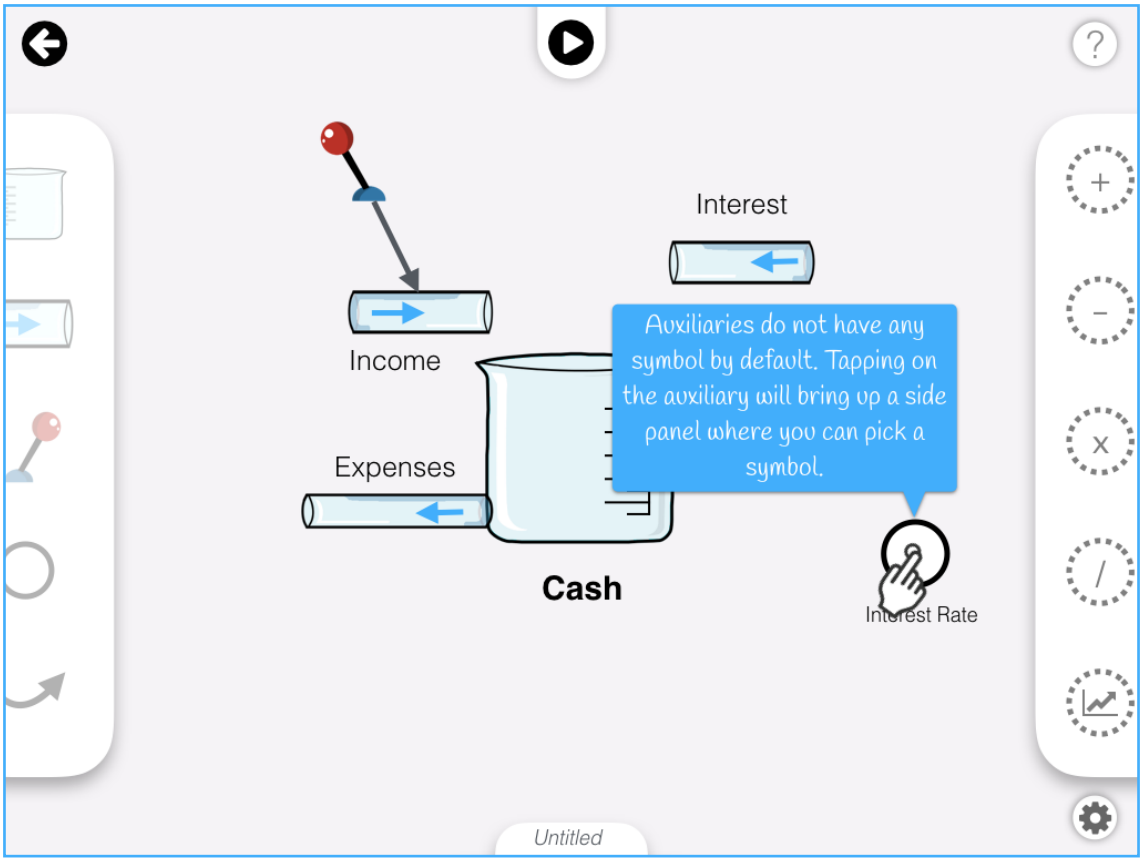


Fig. 3.1.8: Adding a symbol for Interest Rate

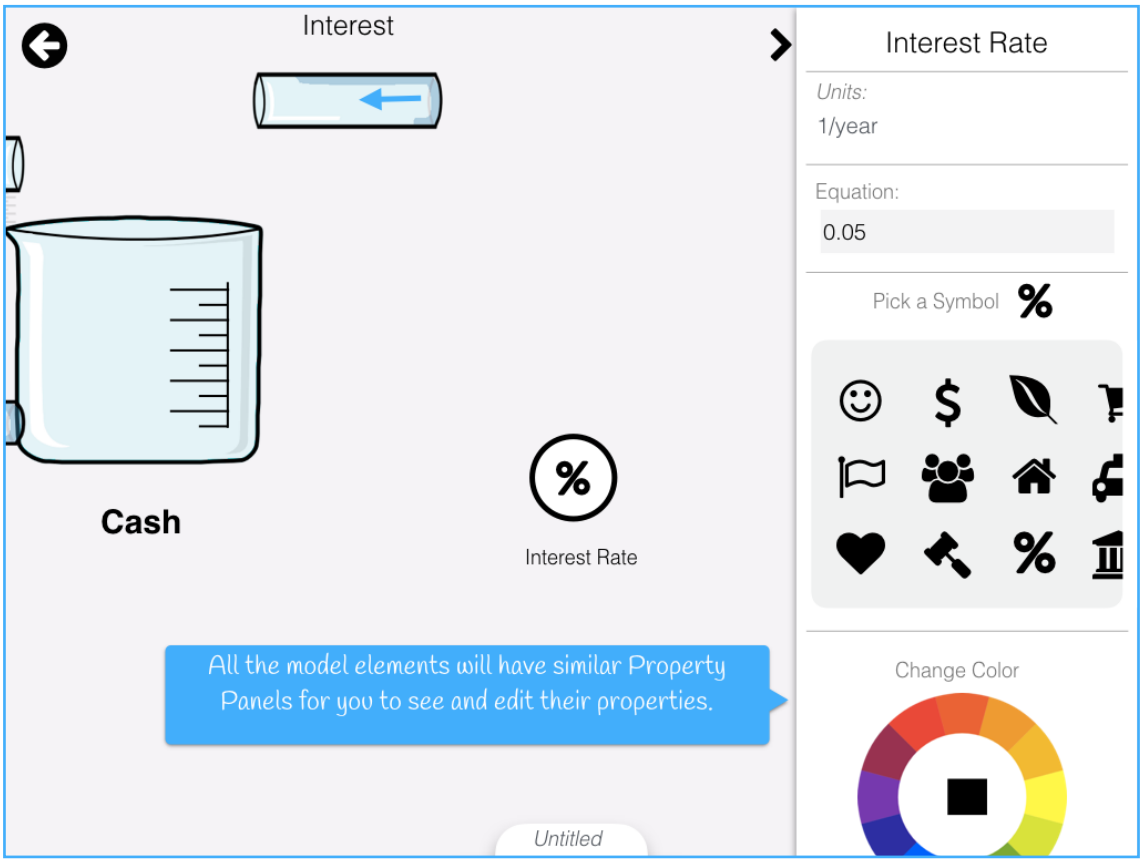


Figure 3.1.9 shows the bank account model with all its stocks, flows, auxiliaries and control elements in place, but with no mathematical operators. Even in this incomplete state, the model can be simulated without a problem. This is because, in Splash, every element you add onto the canvas will come with default values. For example, stocks may have a default initial value of zero (i.e. empty containers), flows may have a default value of 10 and auxiliaries may have default values of unity.

To add in mathematical operations to the model, you'll tap on the operator you want from the toolbar and then tap on the canvas to add it in. Figure 3.1.10 shows a division operator being created. The operator will be created without any input values. To specify the inputs, you'll have to connect the variables that you want to divide to the division operator (refer Figure 3.1.11). The order of the operation will be determined by which variable you connect first and which second. Notice that in Figure 3.1.11, the order of the division operation is actually not how it should be. It ought to be 'Cash/Average Lifetime' but it's the other way around. In case the order of any mathematical operation isn't how you want it, you'll simply double-tap on the operator element to reverse it (see Figure 3.1.12). The output from the operator can be connected to a flow, an auxiliary or another operator element. In the bank account model, the outputs from the multiplication and division operators are used to drive the Interest and Expenses flows respectively (Figure 3.1.13).

Fig. 3.1.9: Bank account model without math operations

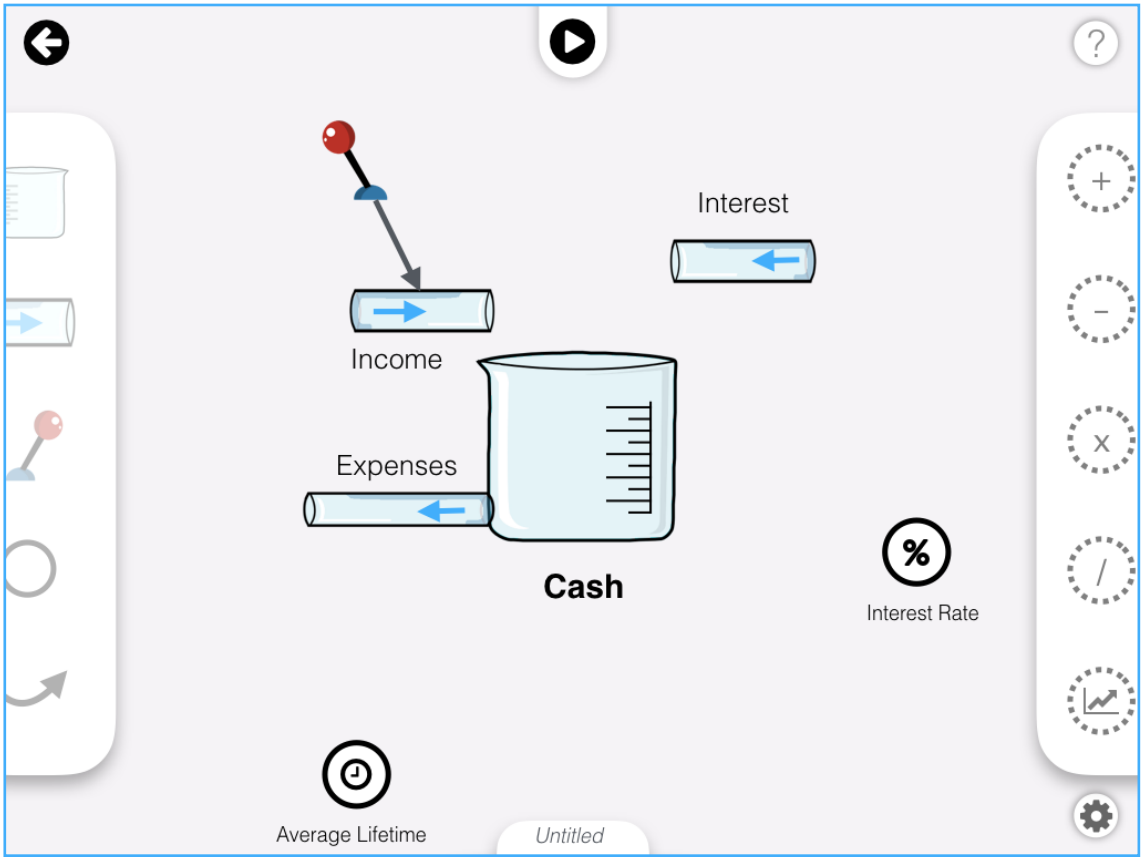


Fig. 3.1.10: Adding a division operation

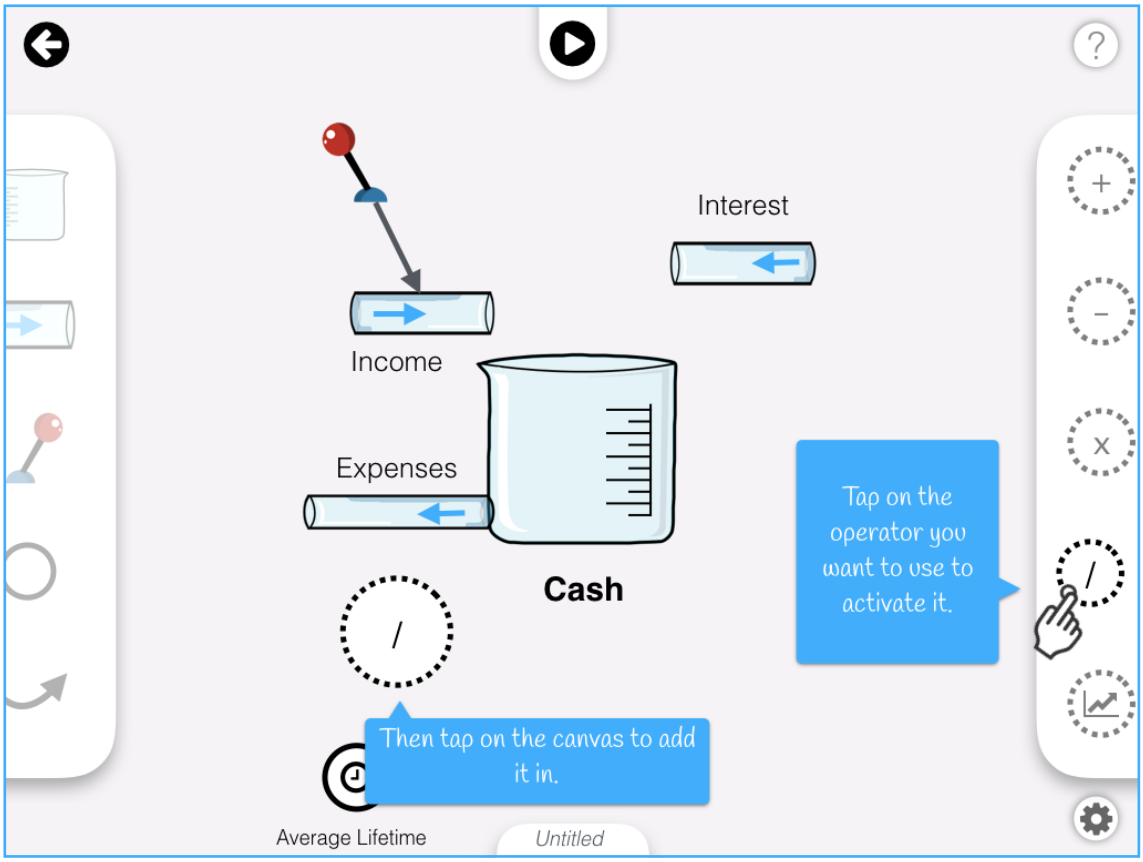


Fig. 3.1.11: Specifying the inputs for division

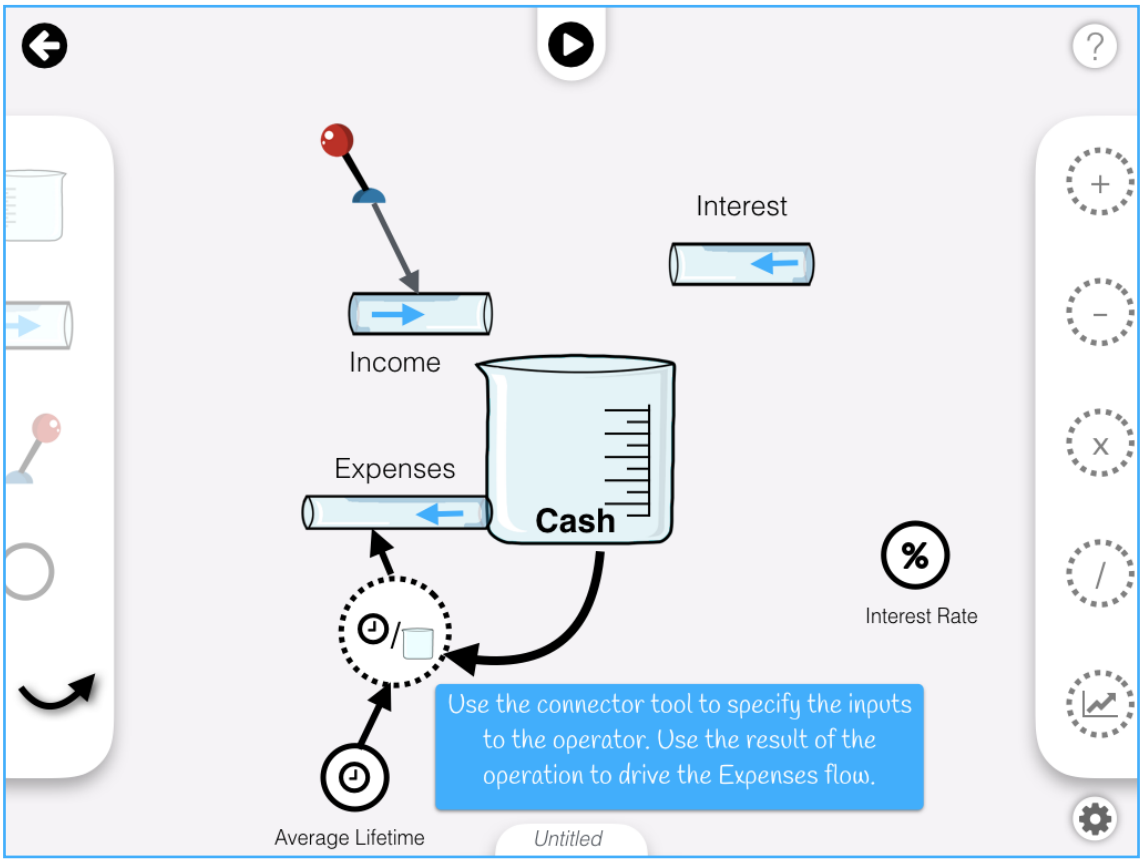


Fig. 3.1.12: Double tap to reverse the equation

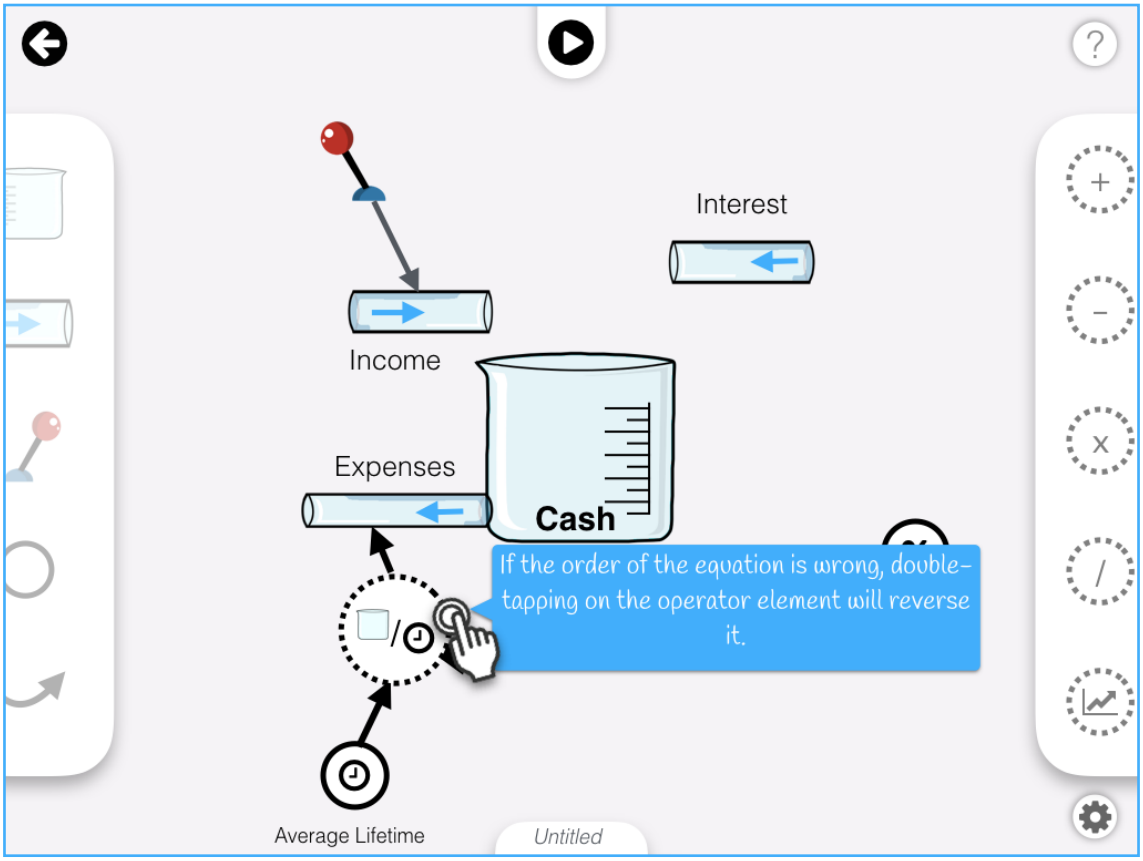
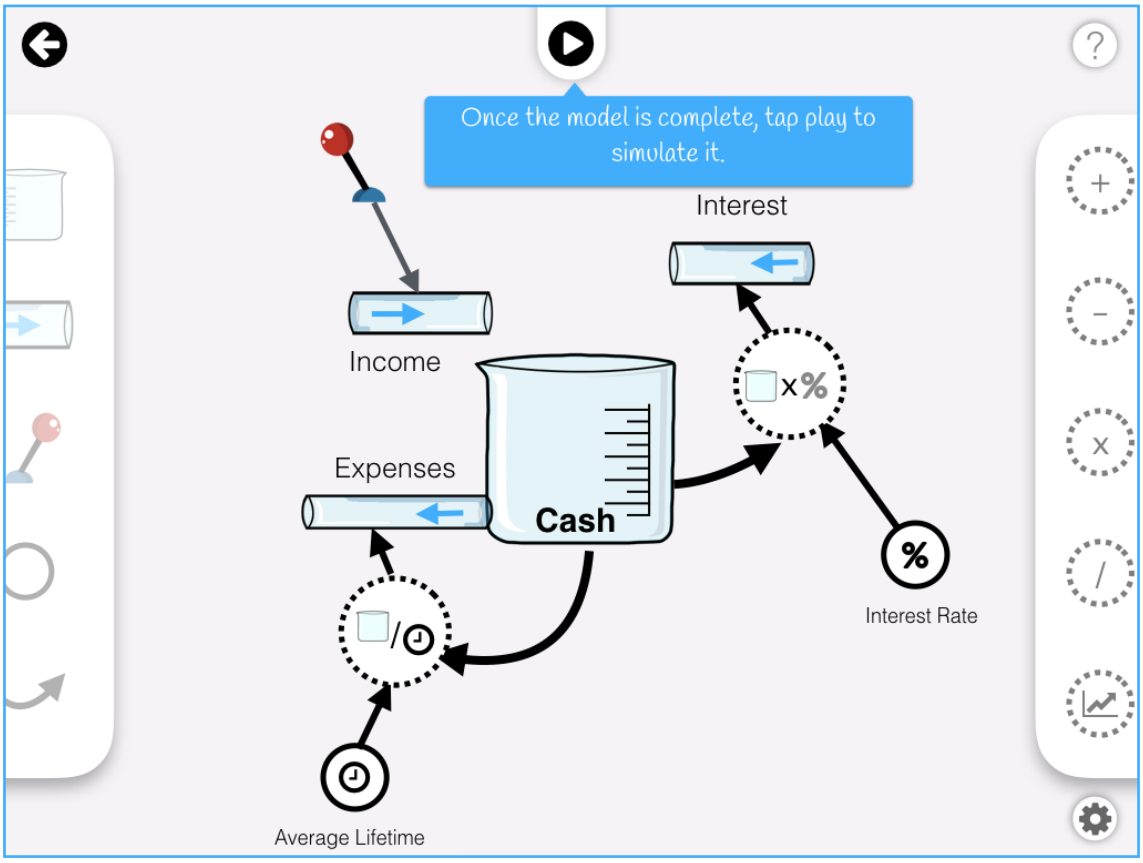


Fig. 3.1.13: The completed bank account model



3.2. Simulate a model

Tapping on the play button at the top of the screen will activate the *Simulation Mode.* The model title and the creation toolbars will disappear and the background color will change (refer Figure 3.2.1). A slider will appear at the bottom of the screen to let you control the simulation speed.

A liquid representing money will start flowing through the Income and Interest pipes, fall under gravity into the Cash container and then flow out through the Expenses outflow. The amount of stuff flowing through the Income pipe will depend on the position of the control element. The amount flowing through the Interest and Expenses pipes will depend on the values of the equations that drive them.

Once the model is in the Simulation Mode, you can no longer edit its structure. You will however, be able to change the inputs of the model while it's running by using the control elements. If there's a switch, you can turn it on and off, if there's a slider you can adjust its value and if there's a lever you can set its position (refer Figure 3.2.2).

In addition to watching the liquid flow and accumulate in the system, you will also be able to judge how the model behaves over time using a set of graphs. In Splash, all the values

Fig. 3.2.1: Simulating the bank account model

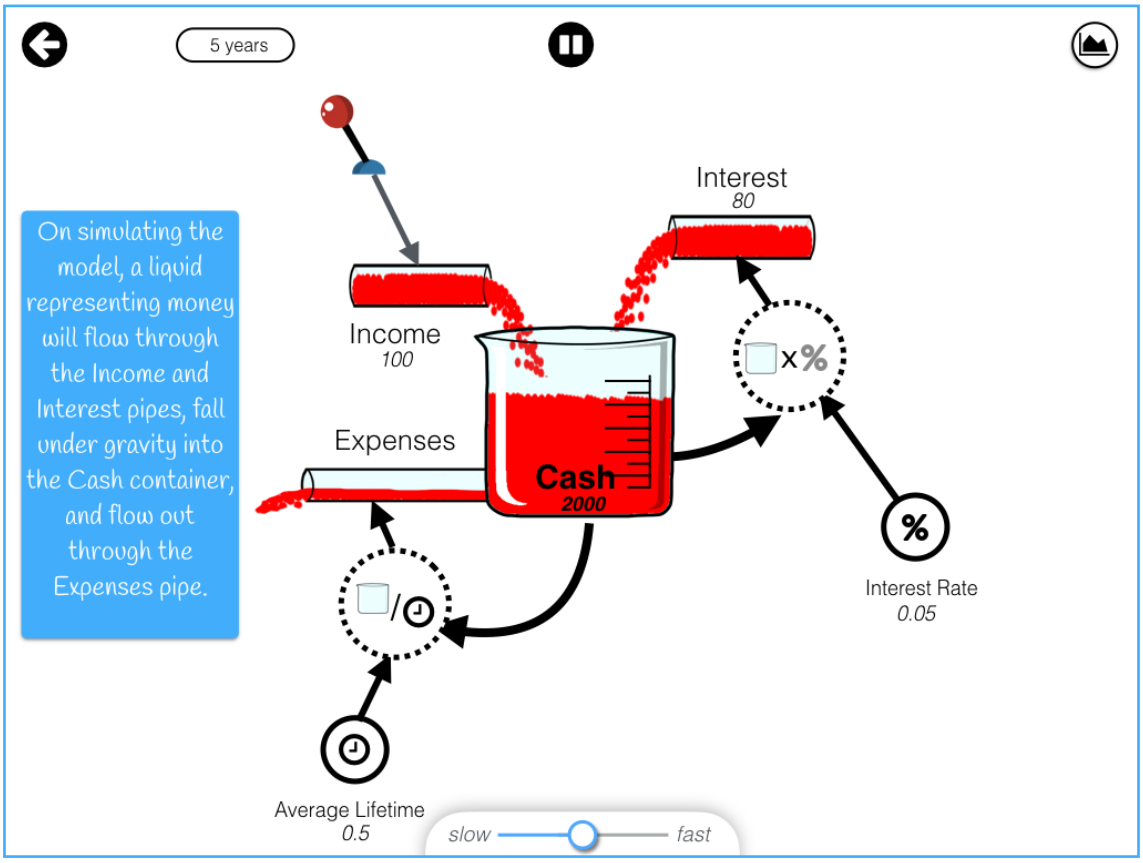
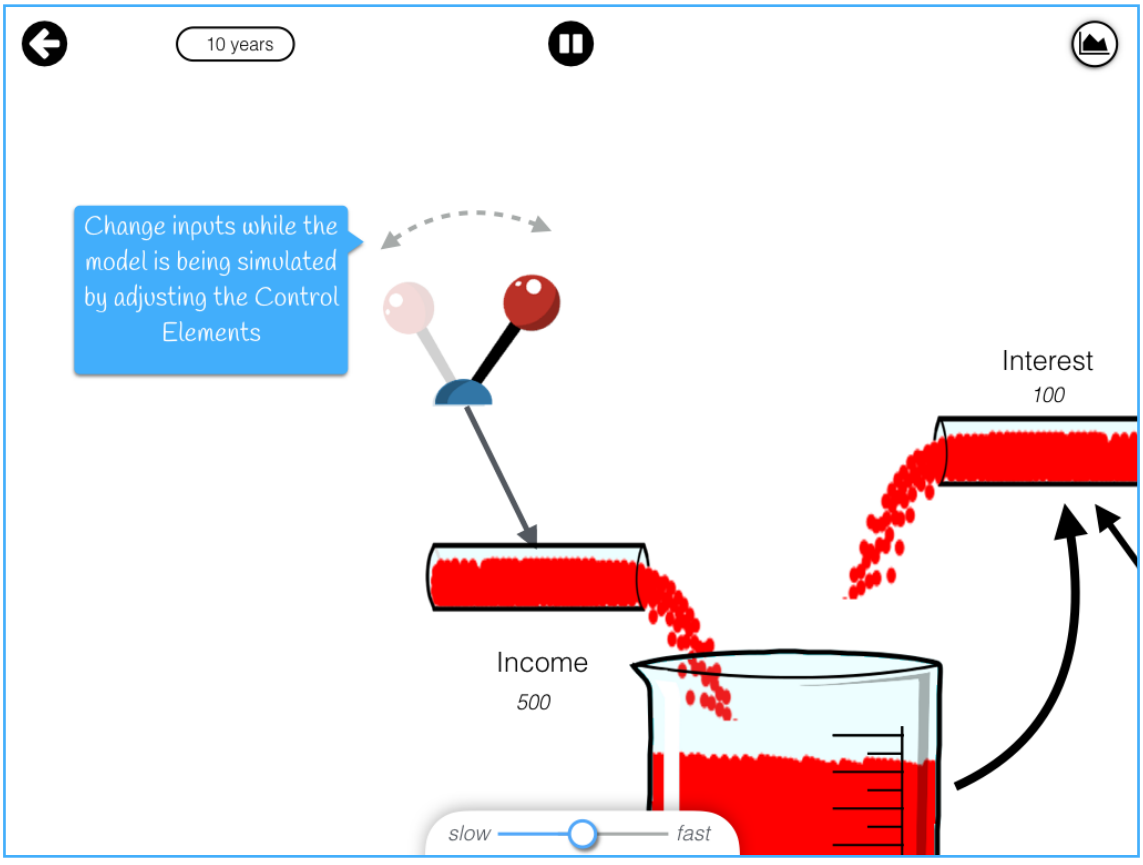


Fig. 3.2.2: Changing inputs at runtime



of all variables in the model will be automatically graphed over time. The plots will be auto-scaled and have a zero origin. Tapping on the plot icon on the upper right corner of the screen while the model is in the Simulation Mode will bring up all the graphs (refer Figure 3.2.3). The graphs will be arranged in a vertically scrolling side panel. To rearrange the order of the graphs, you'll hold down a graph to select it, and then drag it to where you'd like it to be. If you drop a selected graph onto another, the two of them will merge together (Figure 3.2.4, 3.2.5). Their scales and colors will automatically get adjusted and a legend will show up to clarify what's what. If you want to take a closer look at the plots, tapping on a graph will show it in an expanded view (Figure. 3.2.6)

Fig. 3.2.3: Behavior over time graphs

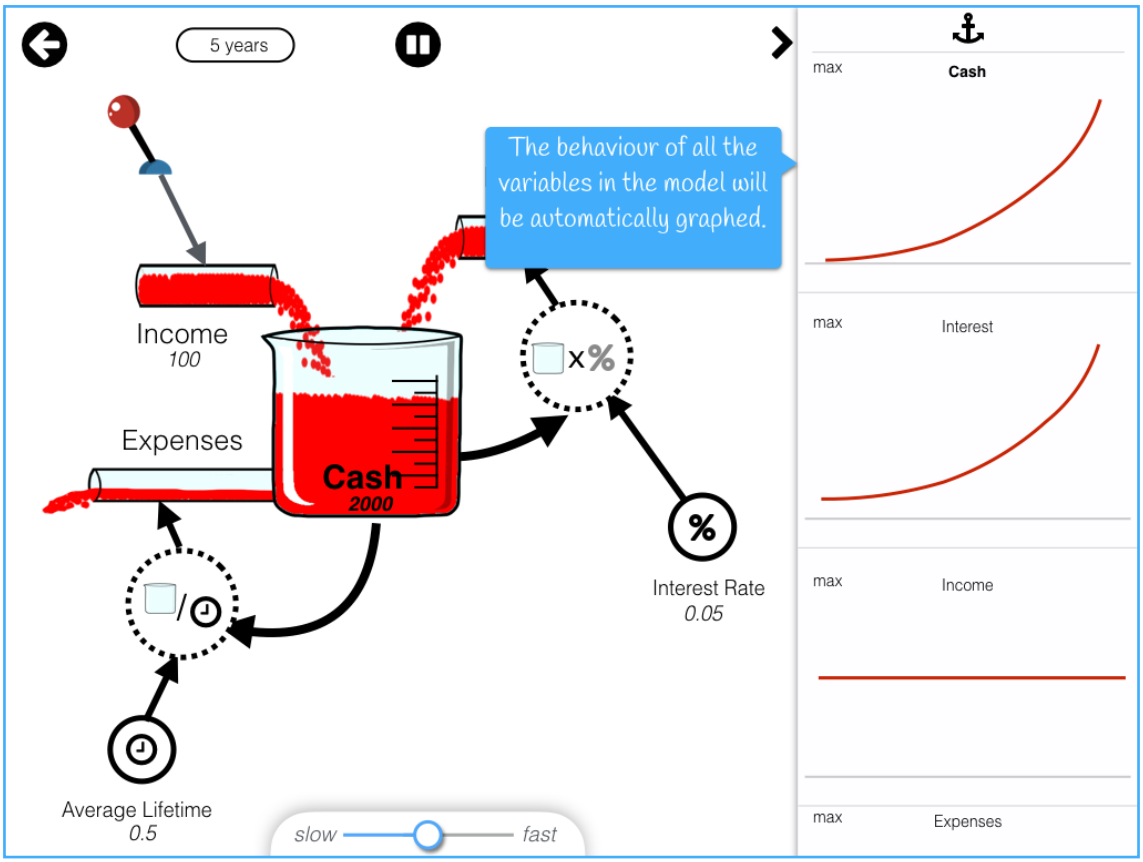


Fig. 3.2.4: Drag graphs to rearrange or merge them

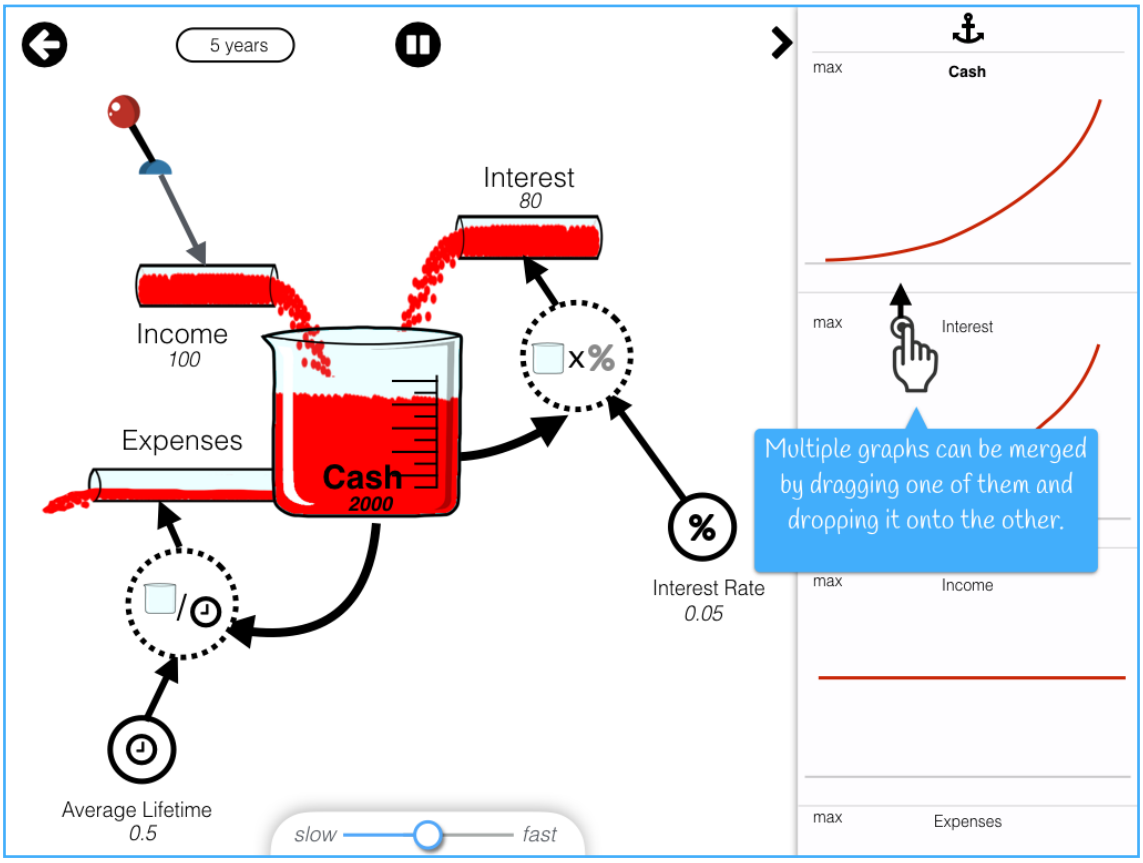


Fig. 3.2.5: Merged Cash and Interest plots

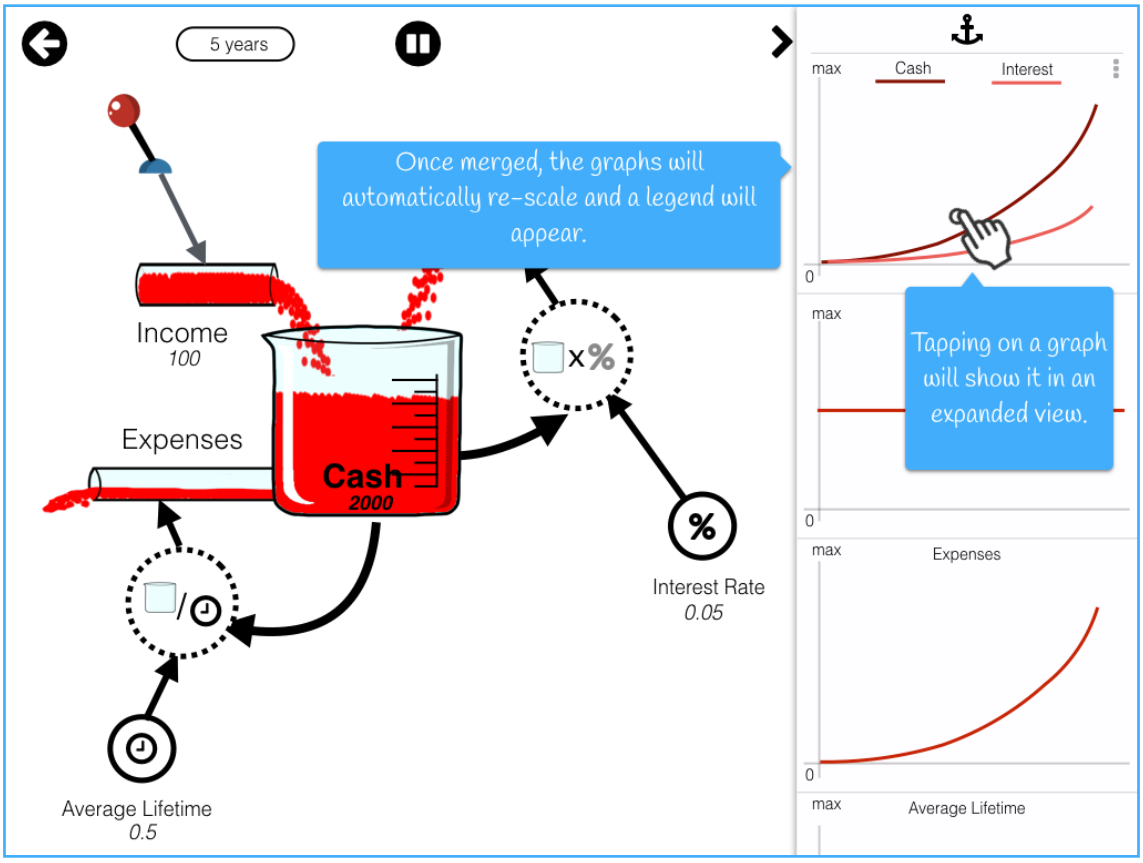
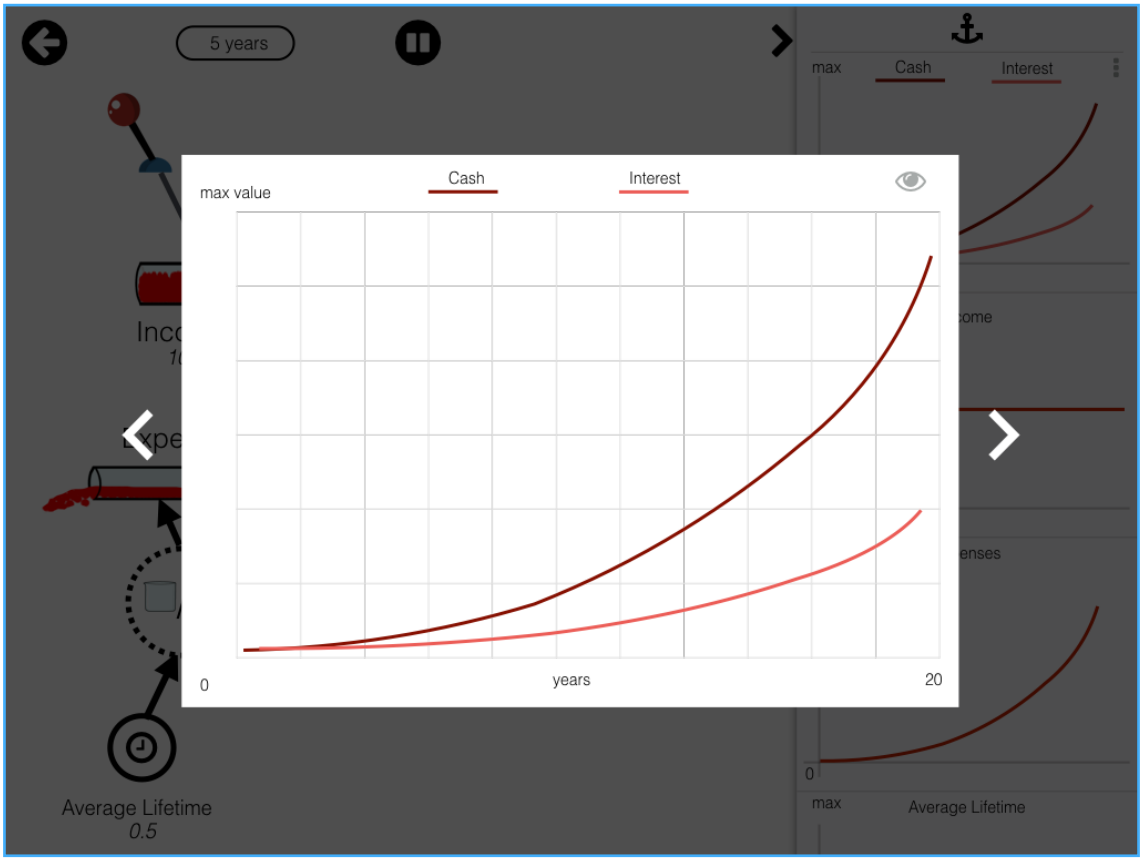


Fig. 3.2.6: Seeing graphs in the expanded view



3.3. Import and duplicate an existing model

Splash will be built with the ability to import models from Google Drive[2]. To do this, you'll tap on the cloud download icon from the Model Library (Figure 3.3.1). If you haven't already signed into Google Drive, you'll be prompted to do so. After signing in, you can find, select and import the Splash model files that you want to download from your Drive. Once the file has been imported, it'll show up in your Model Library.

At times, you may want to duplicate a model and work on the copy. For example, consider that you've imported the 'CO2 Model' that's shown in your library in Figure 3.3.2. Tapping and holding on the CO2 Model will select it and a tiny menu will pop-up on the bottom of the screen (Figure 3.3.3). Tapping on 'Duplicate' in this menu will create a copy of the CO2 Model, and will ask you to give the copy a meaningful name (Figure 3.3.4). Let's say you call the copy 'Carbon Cycle'. Once you're done, it'll show up as a separate file in your Model Library (Figure 3.3.5).

3.4. Edit a model and compare runs

[2] The first version of Splash will support imports only from Google Drive. Imports from email and other cloud storage services may be included in future design iterations.

Fig. 3.3.1: Importing models from Google Drive

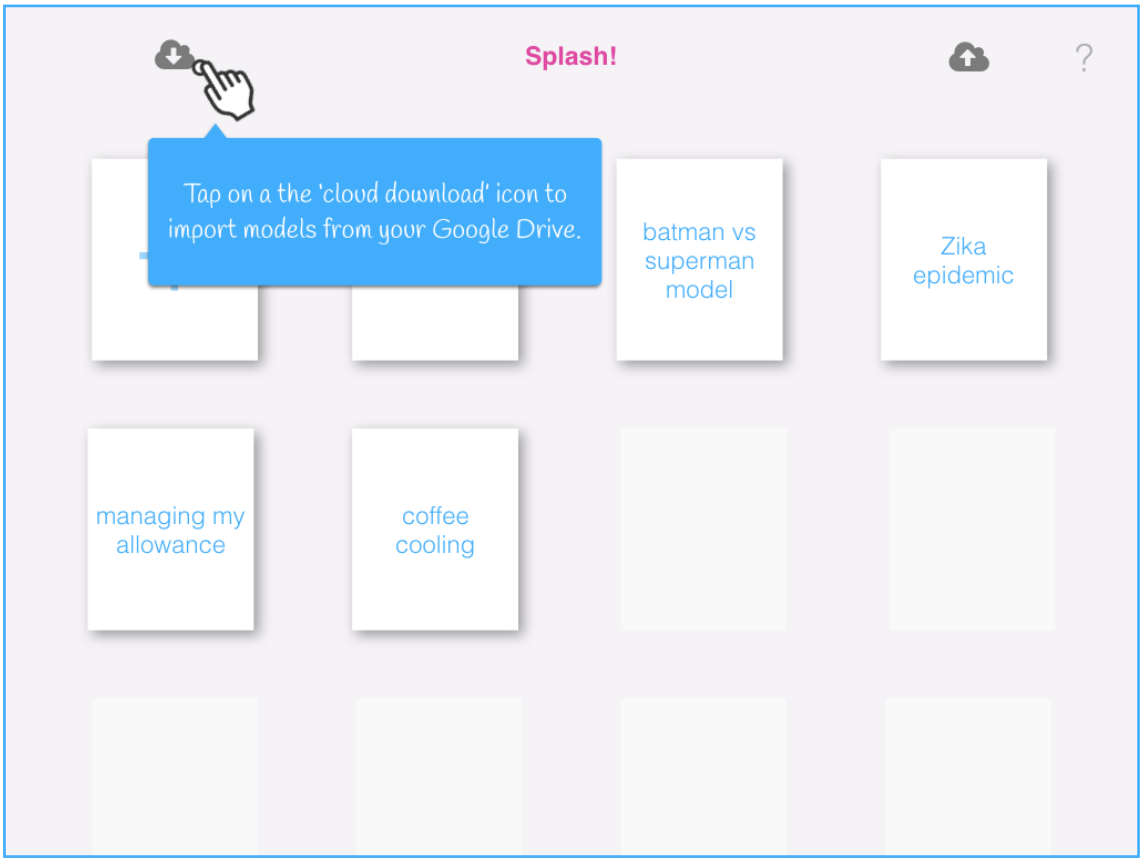


Fig. 3.3.2: Selecting a model

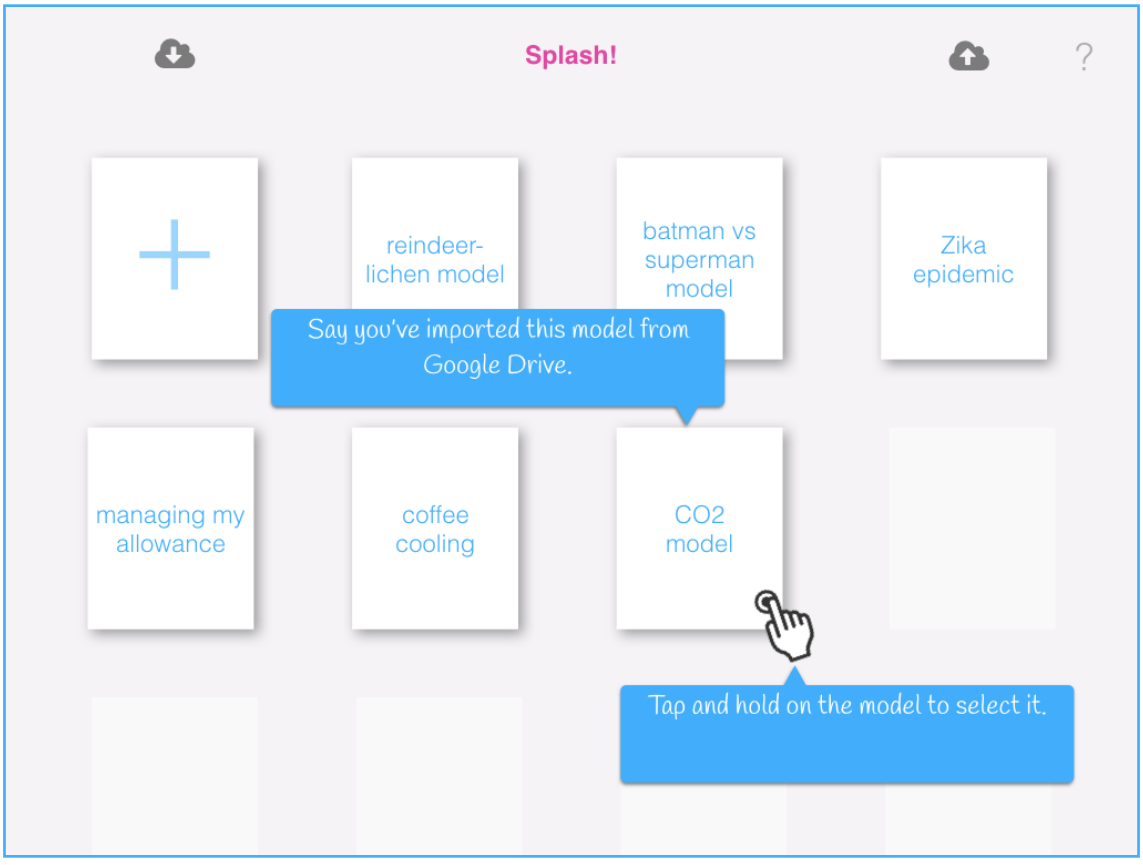


Fig. 3.3.3: Duplicating a model

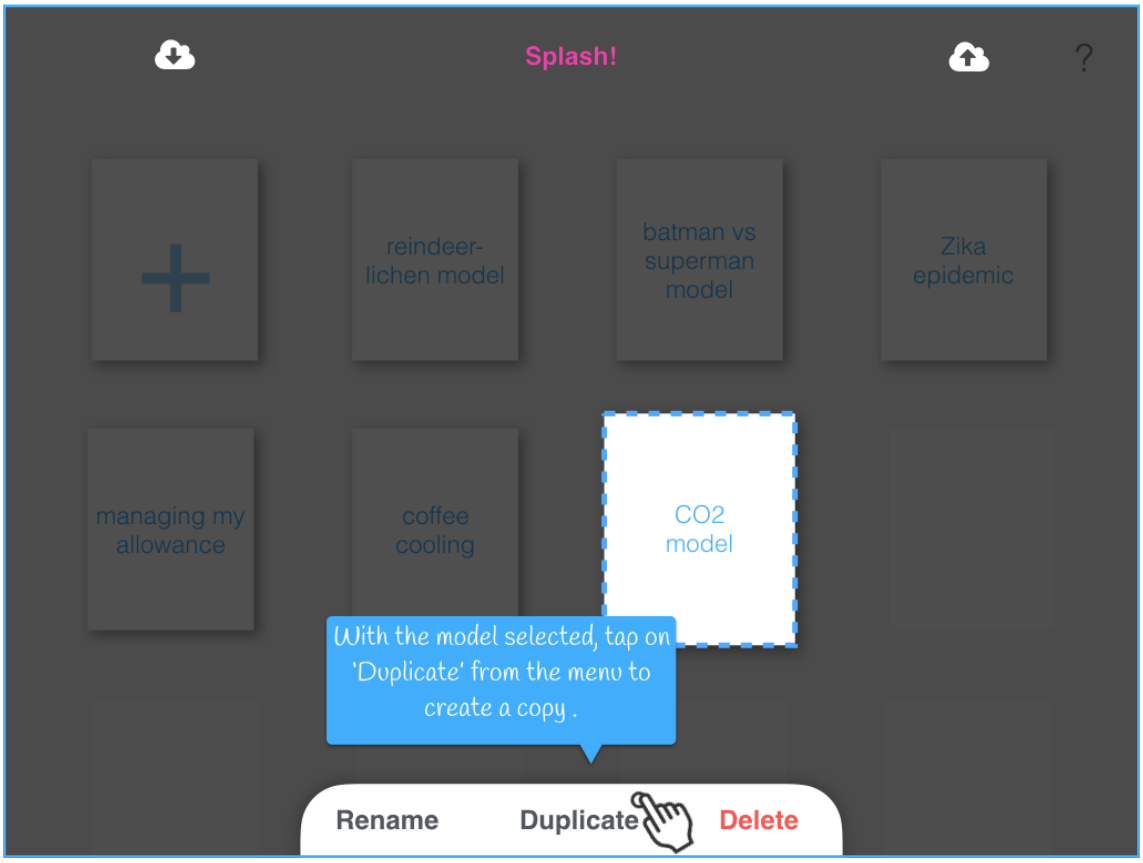


Fig. 3.3.4: Naming the duplicate model

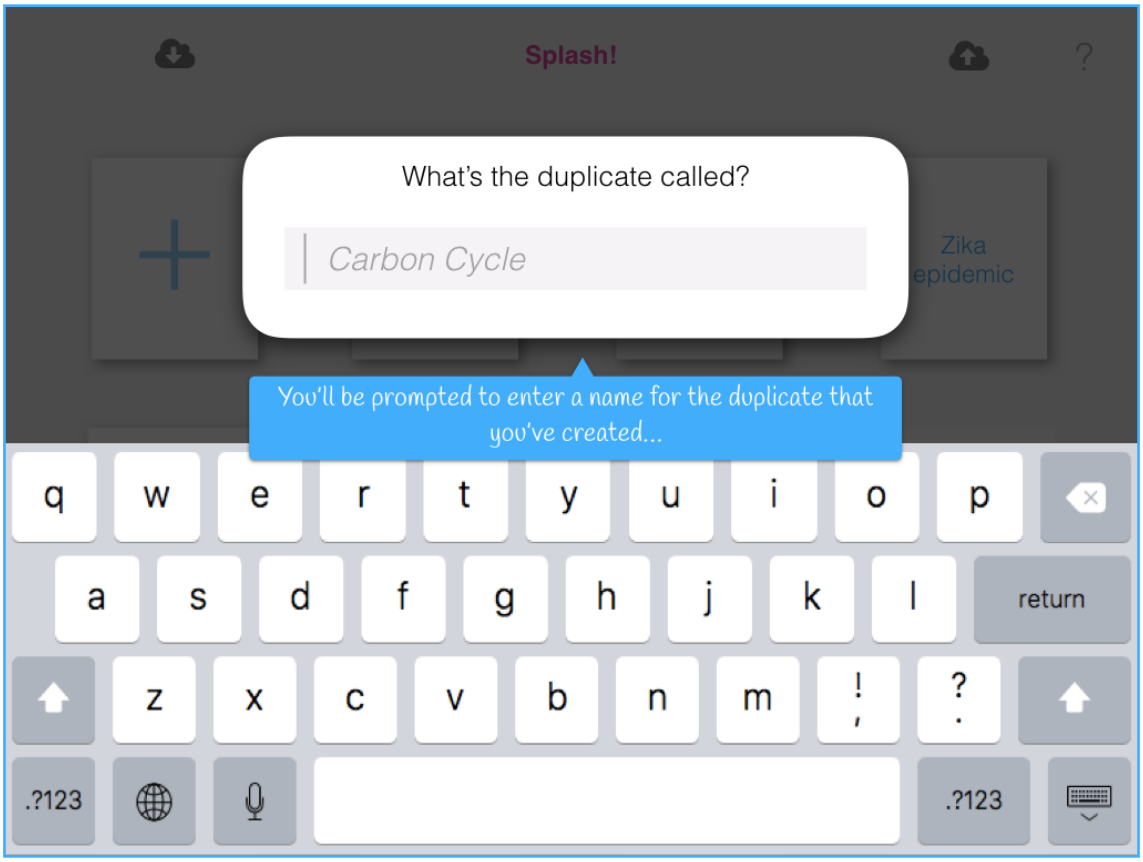


Fig. 3.3.5: Duplicate created in the Model Library

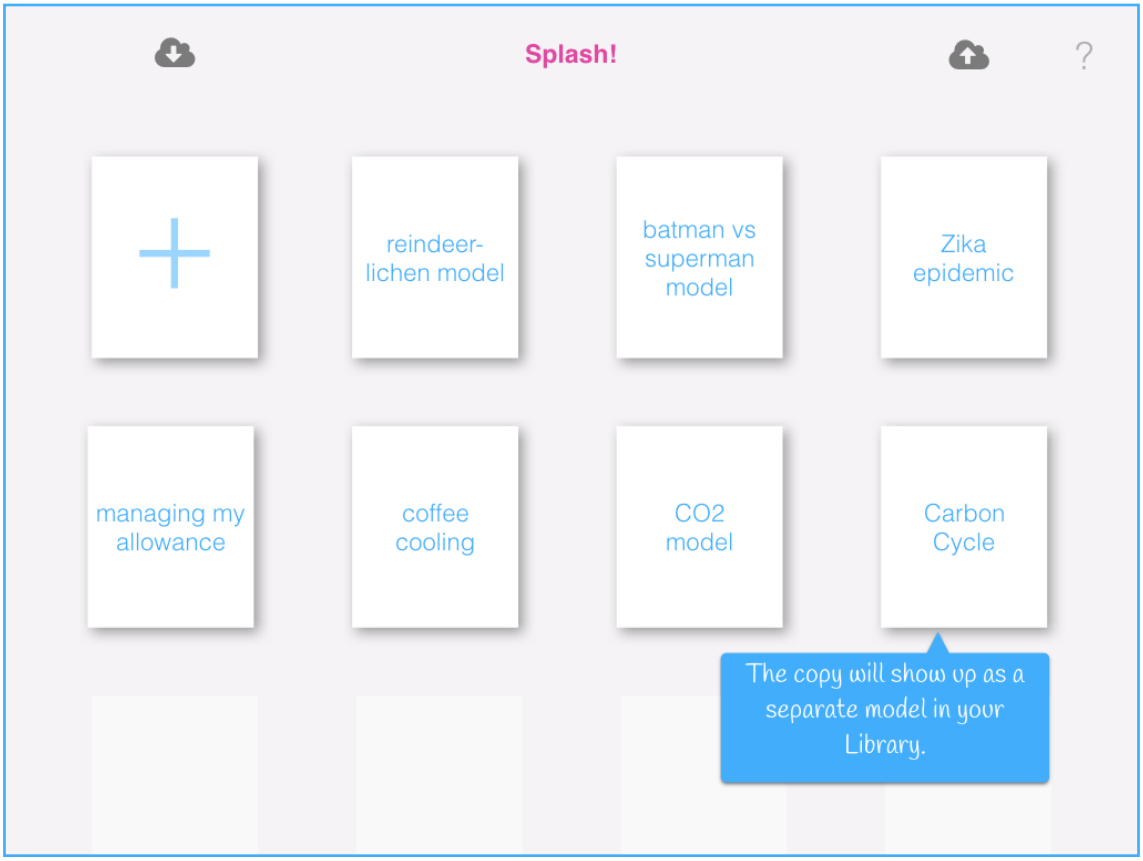
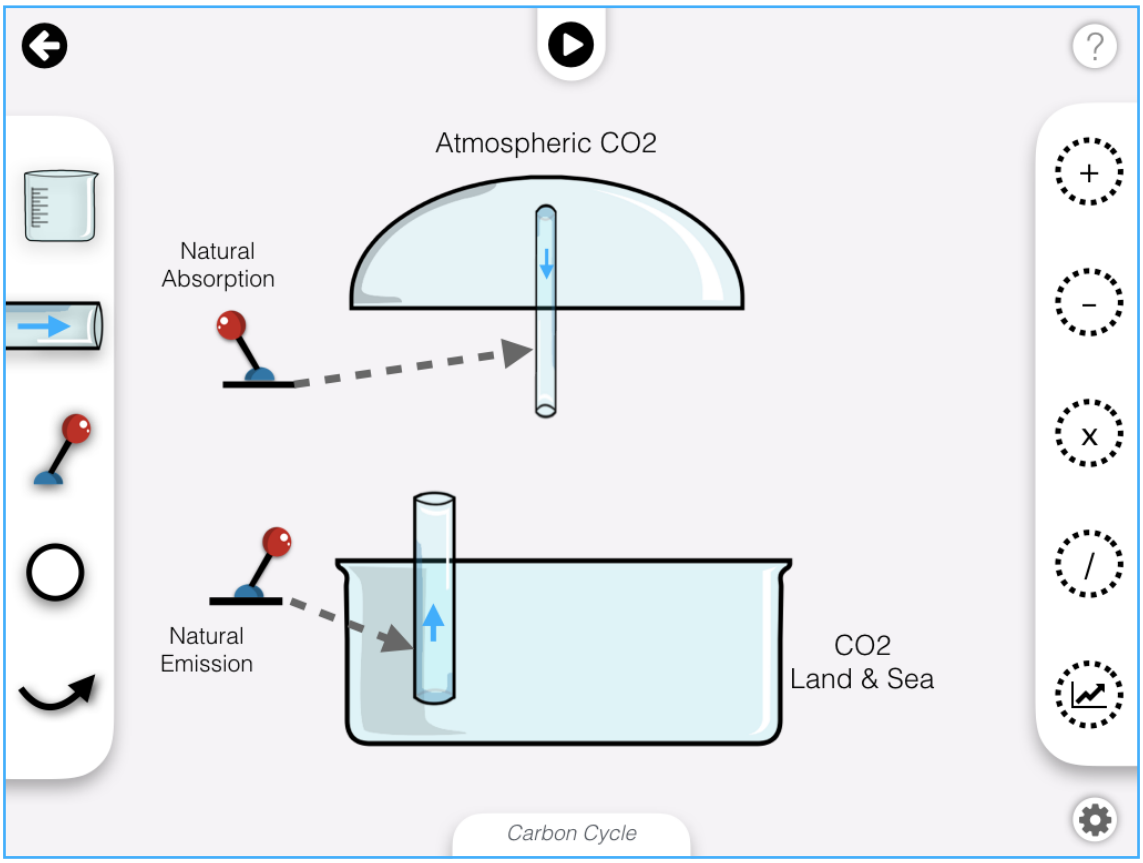


Fig. 3.4.1: The imported Carbon Cycle model

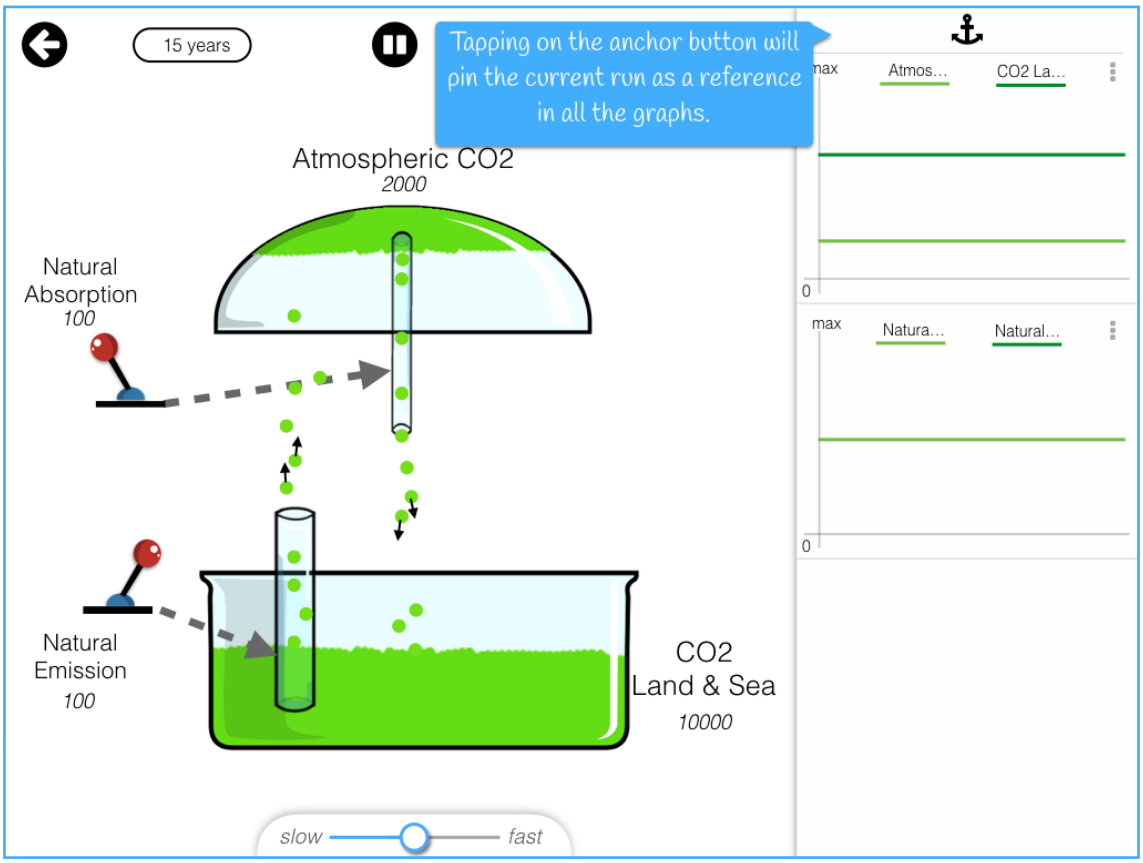


Say you now want to edit the Carbon Cycle model that you duplicated in section 3.3. Figure 3.4.1 shows the model on the Creation Canvas. The model has flows for natural CO2 emissions and absorption, but does not have a flow representing anthropogenic emissions. You want to add that flow into the model along with a lever to control it while the model is being simulated. But first, you want to save the behavior of the current model so you can use it as a reference later on.

In Splash, you will be able to pin a run to the graphs. To do this, you'll simulate the model and tap on the anchor icon located at the top of the graph panel (Figure 3.4.2). Doing this will pin the current run as a reference in all the plots. You won't see any change in the graphs immediately though, since the current run will exactly match the reference. You will however, see the anchor button turn from gray to green, indicating that the run has been saved.

Returning to the Creation Canvas, you'll add in a flow, call it 'Anthropogenic Emissions'

Fig. 3.4.2: Anchoring a reference run in the graphs



and connect a lever to control it (Figure 3.4.3). To specify the default, minimum and maximum values of the lever, you'll tap on it to bring up its Property Panel (Figure 3.4.4).

Fig. 3.4.3: Modifying the model structure

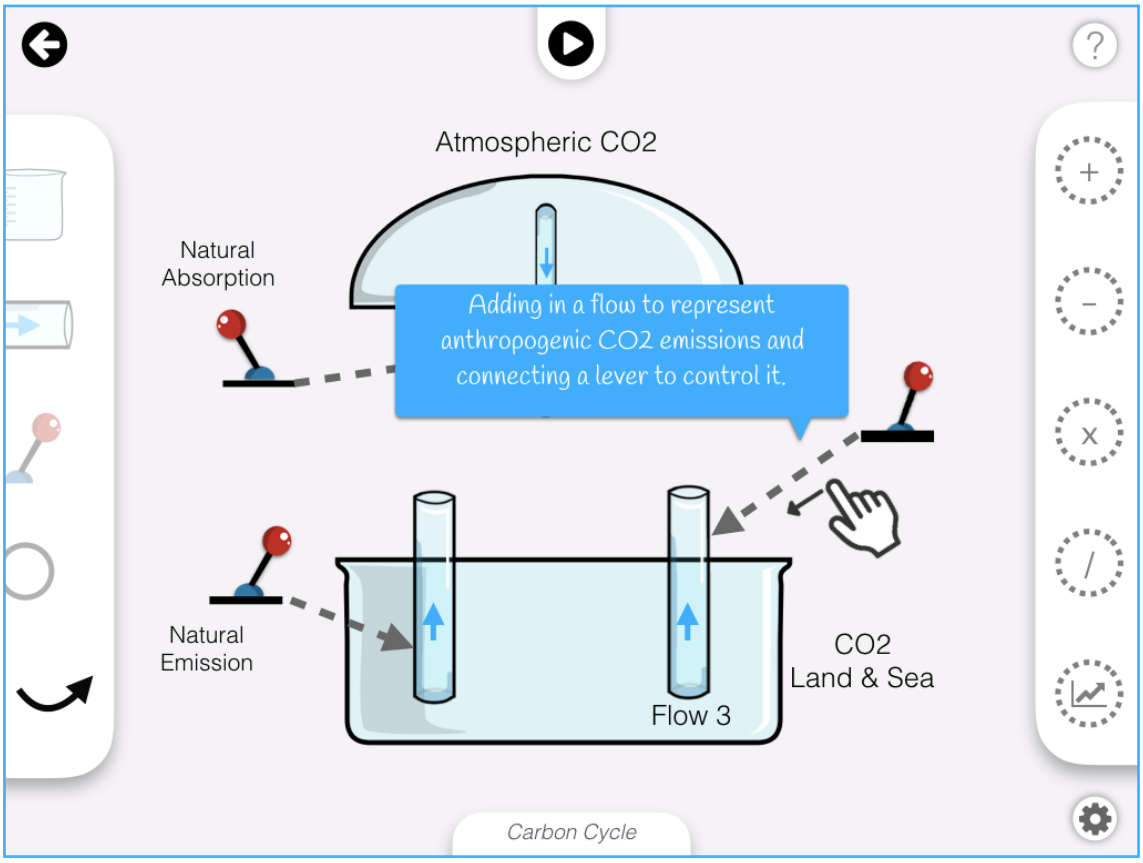


Fig. 3.4.4: Adjusting the lever properties

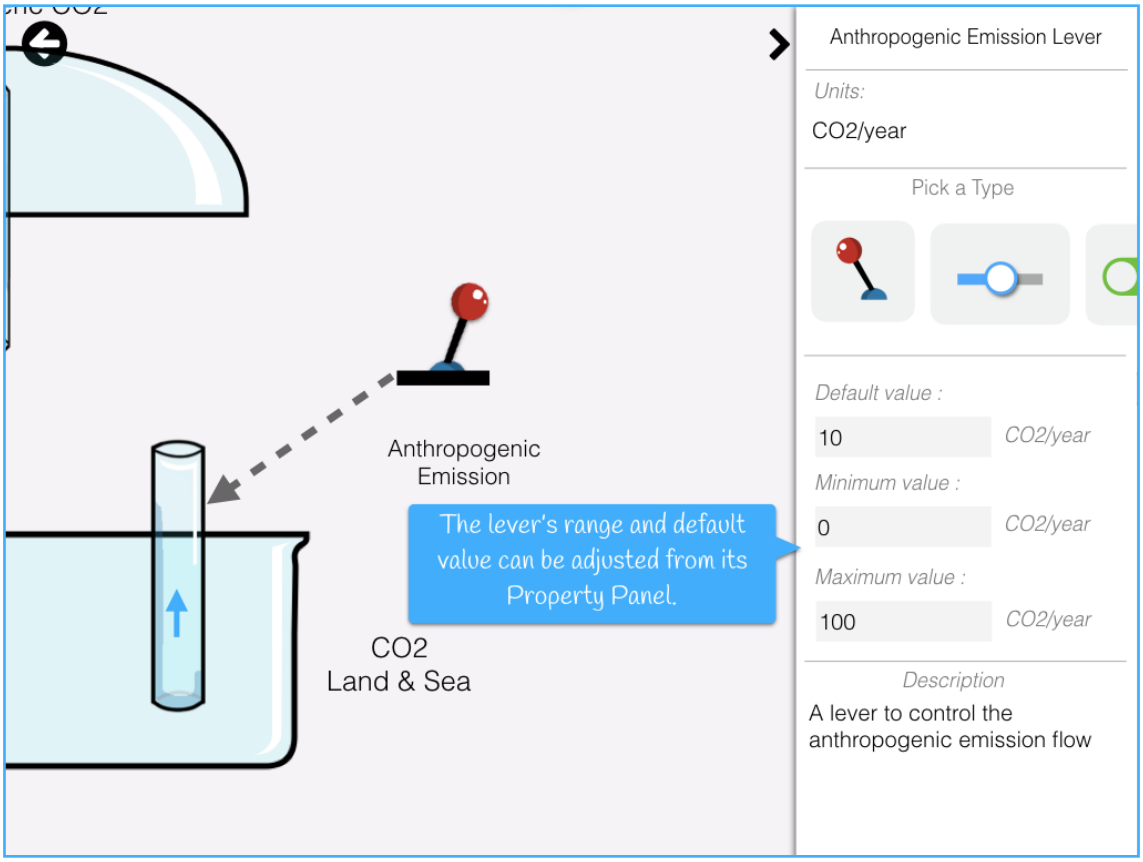


Fig. 3.4.5: Simulating the modified model

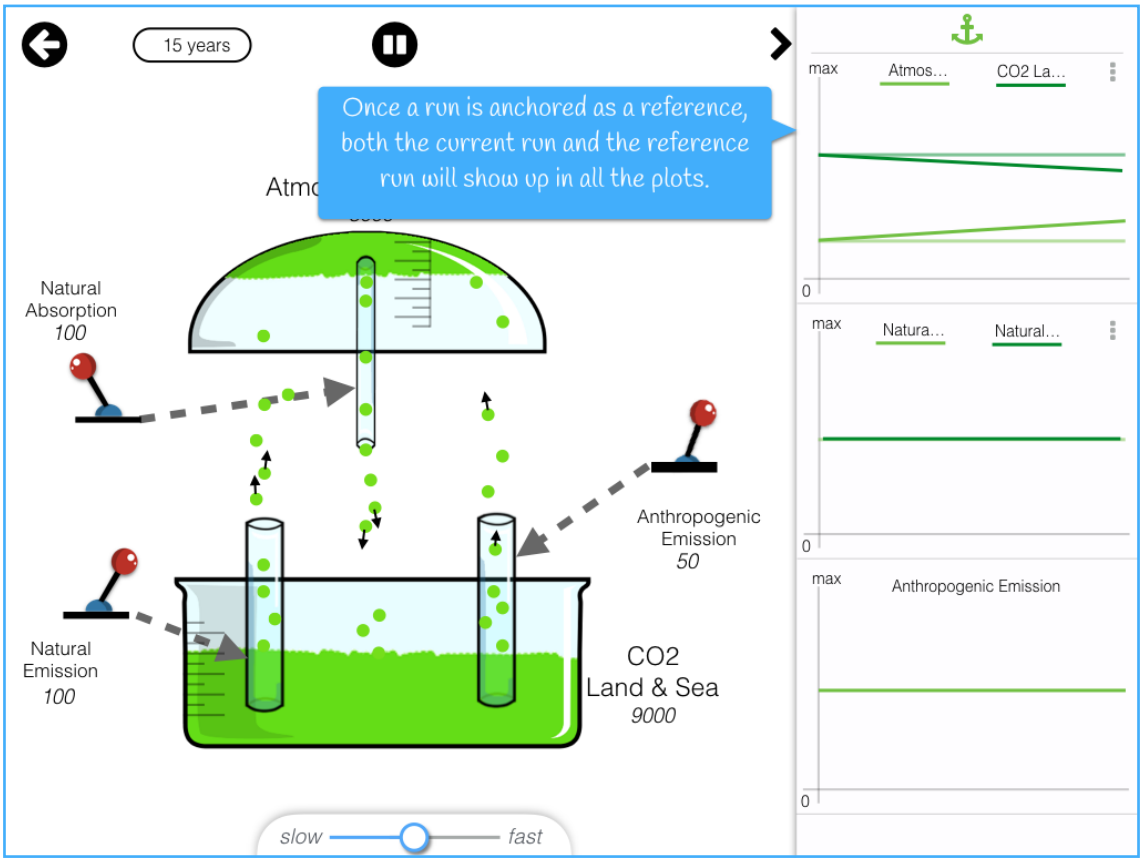


Fig. 3.4.6: Graphs in the expanded view

With the necessary changes made, you'll press play once again to simulate the modified model. This time all the plots will show both the current and the reference behavior (Figure 3.4.5). This way, you'll conveniently be able to compare and contrast different scenarios against your selected baseline. If you want to take a closer look, tapping on any graph will open it in an expanded view (Figure 3.4.6).

3.5. Manage all my models

Splash is designed for tablets and mobile devices. All your model files will be stored on your device and will be accessible through the Splash app. The Model Library is where you'll get to view and manage all the models on your device. The Library will list the models in a scrollable grid of files. The grid will keep expanding as you create and add more models. You've already seen how you might use the Library to duplicate an existing model (Section 3.3). In a similar manner, you'll be able to select a model to rename or delete as well (refer Figure 3.5.1 and 3.5.2).

Once you have a sizable number of models in your Library you'll probably want some way to quickly filter through them and find the ones you want. To do this, you'll drag the Model Library downwards to reveal a search bar and sorting toggles (Figure 3.5.3 and 3.5.4). Using the search bar, you will be able to find models by their name (Figure 3.5.5). The sorting toggles will allow you to list the models in alphabetical order or by the date they were last used.

Just like you can import a model from Google Drive, you'll also be able to export your models to Drive using the Model Library. You'll do this by selecting the models you want to export and then tapping on the 'cloud upload' icon in the Model Library.

Fig. 3.5.1: Selecting ‘managing my allowance’

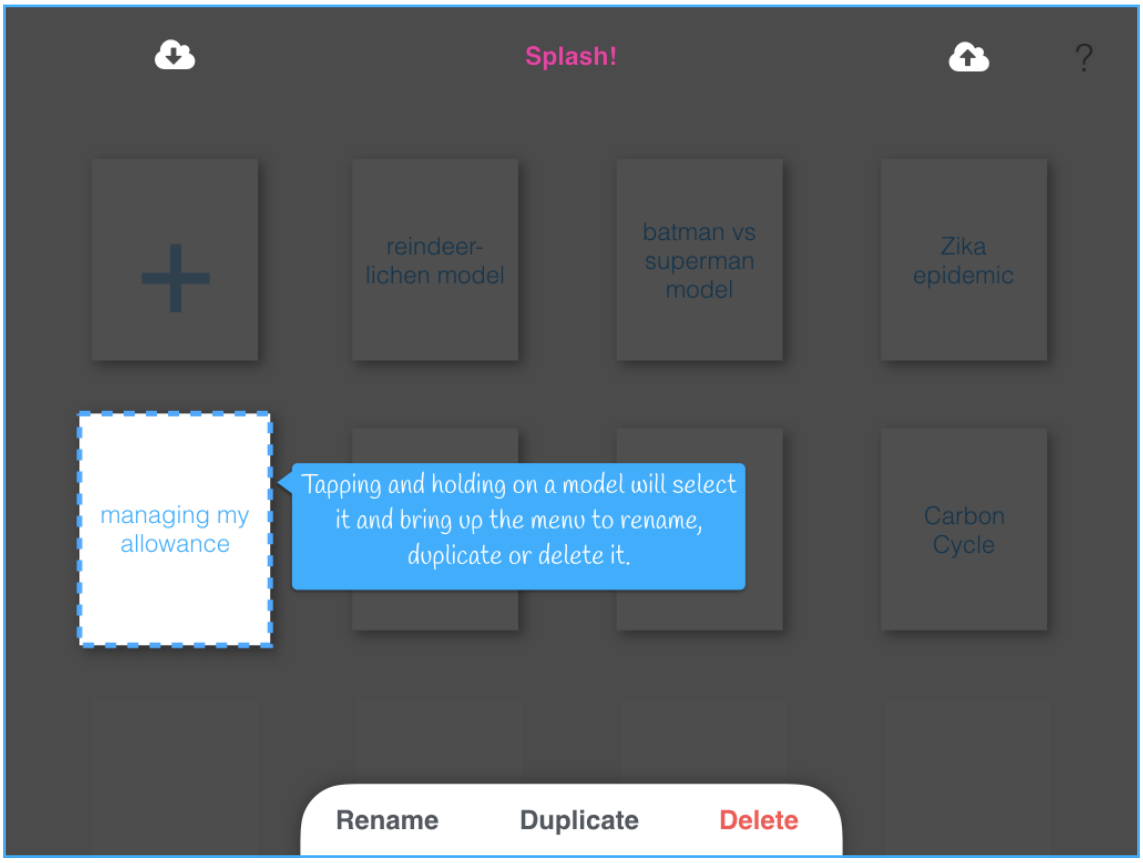


Fig. 3.5.2: Renaming ‘managing my allowance’

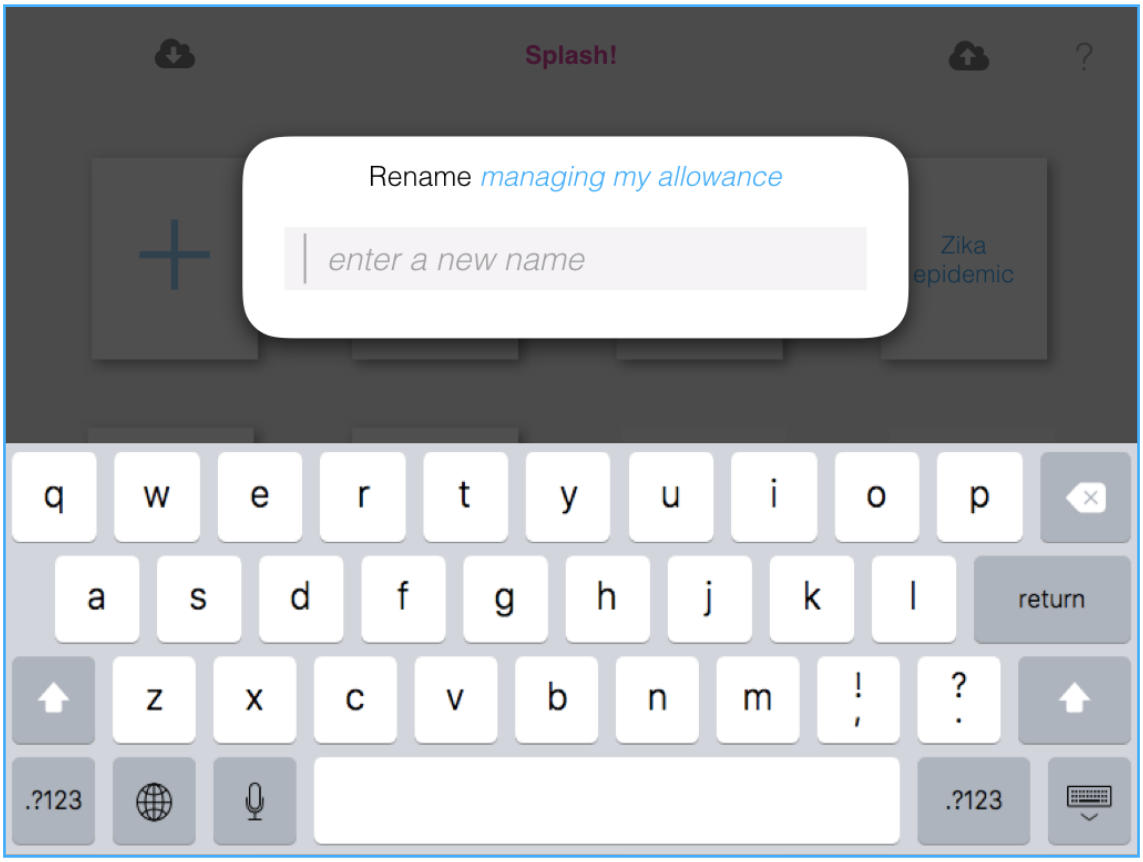


Fig. 3.5.3: Revealing the model filtering options

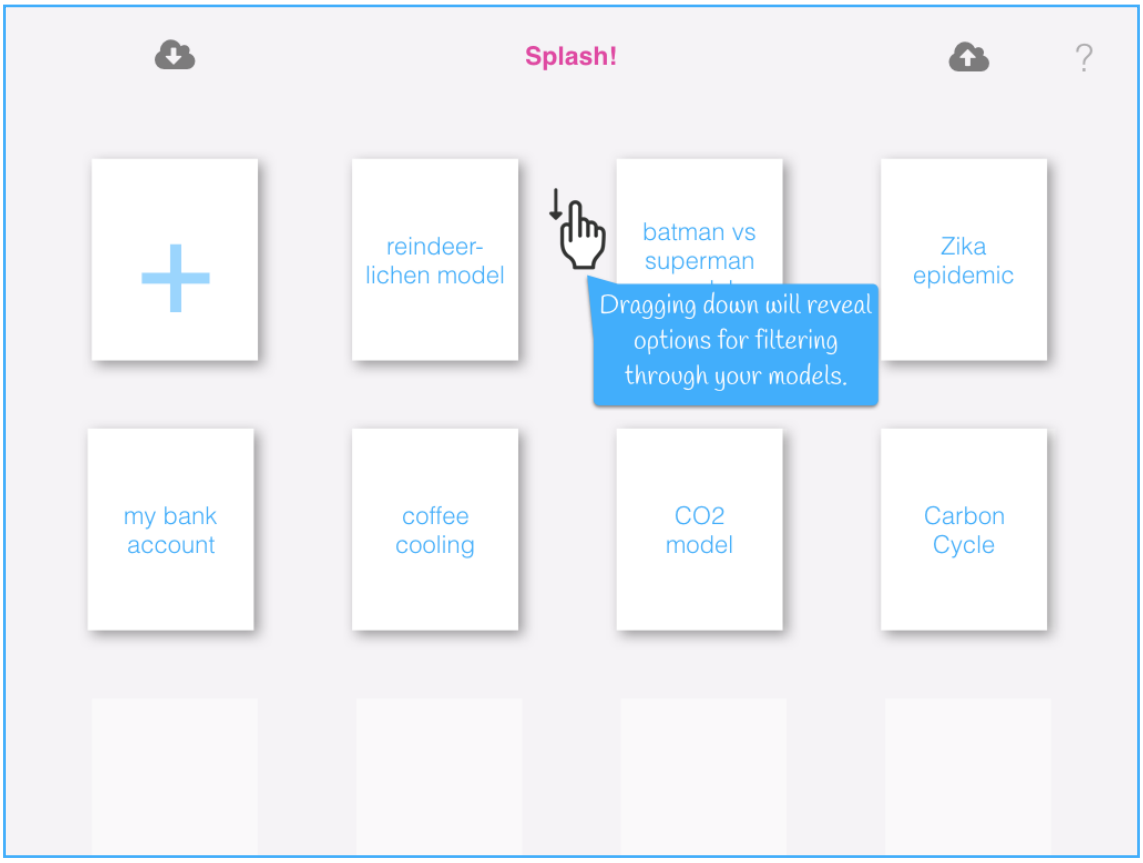


Fig. 3.5.4: The search bar and sorting toggles

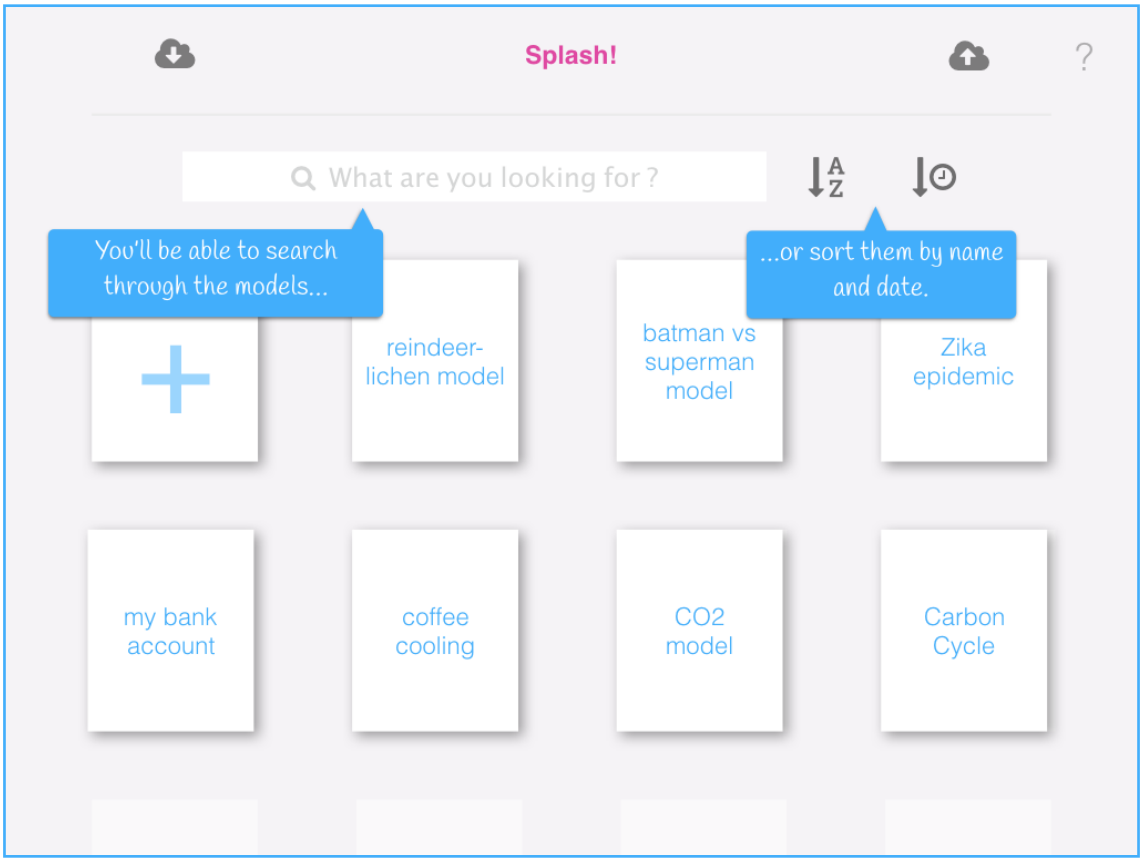
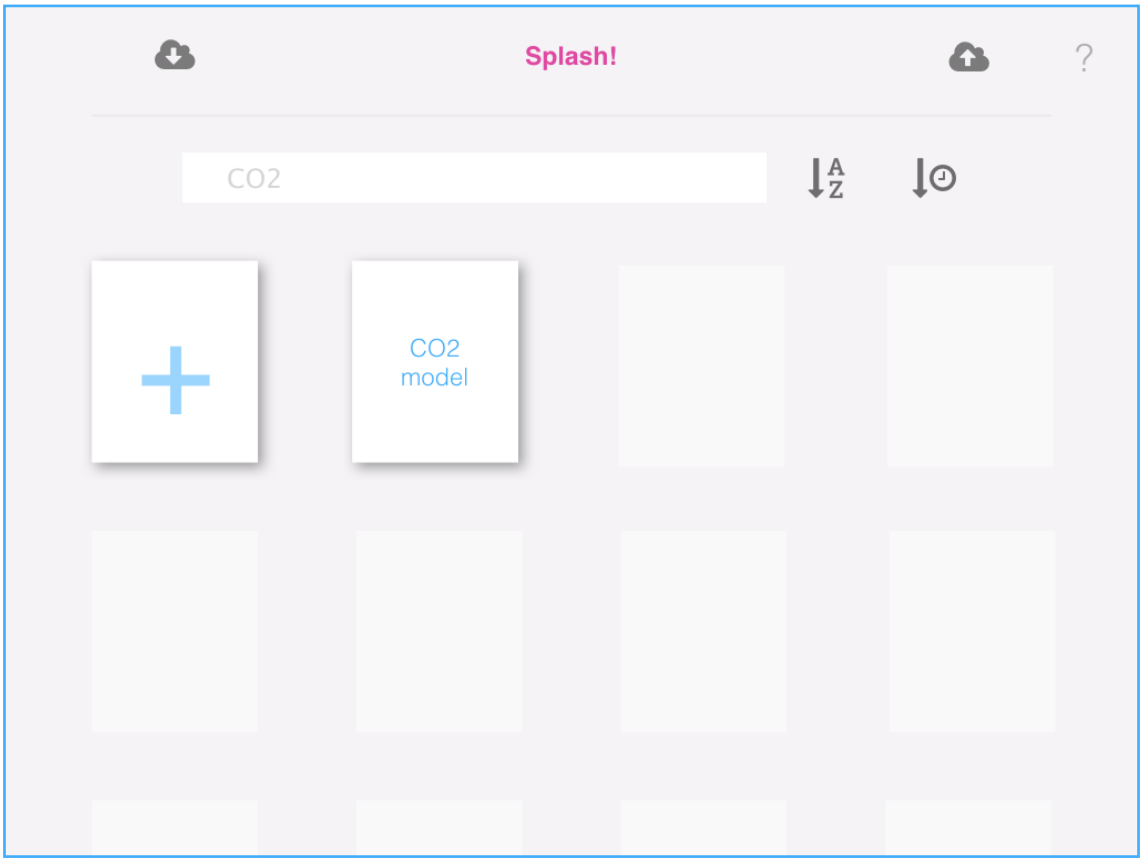


Fig. 3.5.5: Search results for ‘CO2’



4. Scope for Improvement

Like any product, the design of Splash will be an ongoing project well beyond the development and release of the initial version. Given this, it’s important to highlight areas where there is scope to improve our design. Evaluating the final design for Splash against the Design Framework that we had adopted at the beginning of the process helped reveal its strengths and weaknesses (see ***Behind the Scenes*** for details of the Design Framework). This evaluation was carried out by the design team and is based on their intuitive, qualitative assessments of the final design.

First off, we assessed how well Splash supports the use and learning of key aspects of system dynamics like accumulations, endogeneity, causality and aggregation. By having liquid physics at its core, Splash drives home the idea that systems are comprised of flows and accumulations. It also underscores the principle of mass conservation. However, the design does not specifically nudge users towards building endogenous models. This can perhaps be improved by having the software automatically evaluate how endogenous a particular model is and accordingly give it an ‘endogeneity rating’. Similarly, Splash’s design lets the user decide what level of aggregation is appropriate and what structure is needed to capture the relevant causality. Having some kind of automatic validity assessment built into the software might help here. But doing this would require Splash to have access to all the necessary domain knowledge for any given model, along with some

form of deep artificial intelligence. Given the amount of effort needed to create such automatic validity assessment, the feature is not included in the current design.

Moving on, we considered whether the UI and UX in Splash is intuitive and easy to use. While the design is likely to be self-evident for the target audience we have in mind (beginning SD learners, school students and teachers), feedback on the design from potential users was not formally sought or received during the design process. Such user perspectives and inputs would help inform the design and ought to be collected.

Next, we judged whether the design of Splash seems engaging. Giving users the ability to add a guided narrative (using text and audio) to any model they create would likely make Splash more engaging. Having inbuilt tutorials and sample models that are relevant would also help.

At the start of the design process, we wanted the software to be flexible enough to create a wide variety of models. The design of Splash however, is most suitable for models that are small or moderate in size. It also does not support non-negative stocks, arrays and mathematical operations aside from basic arithmetic. However, the amount of flexibility the design does provide will likely be sufficient for our target audience.

Dealing with high levels of abstraction is often part and parcel of building system dynamics models. This may be a hurdle to people learning system dynamics. We aimed to design the software to be as relatable and concrete as possible. Splash's design seemingly lies somewhere between the level of abstraction seen in existing SD software and the level of concreteness seen in entertainment media such as movies and games. It may be possible to conceptualize interfaces that are more relatable than what's currently planned for Splash, but not without creating a drop in performance along other dimensions (such as how well the software honors and promotes core SD principles).

A limitation arising from the use of liquid physics in Splash is that it may not be possible to simulate models instantaneously. Increasing the speed at which the model simulates would be doable, but not to a level that provides instantaneous results. This fact may be frustrating to users who want to quickly perform multiple iterations.

Our current design efforts did not cover the audio aspects of Splash's UI and UX. This is certainly an area to which significant thought, effort and user testing should be devoted while the software is being developed.

Splash's design is not very 'social' in nature. It does not include any integration with social media platforms (such as Facebook/Twitter) for users to conveniently share their work or progress. The design also does not allow for multiple users to work on the same model, at the same time, using multiple devices. Adding such functionality into future iterations of the software will allow Splash to be used in participatory modeling exercises.

Finally, and importantly, given that Splash is meant as an educational tool it is critical that teachers find the software truly useful. If school teachers are to adopt and use Splash at any serious scale, the software will have to strongly dovetail with their existing interests, roles and responsibilities. Ways to do this might include (a) creating tutorials, sample models, and modeling exercises that are in line with existing curricula (b) creating a platform where students can conveniently work on modeling assignments from their teachers and where teachers can easily review and provide feedback, and (c) pushing for systems thinking and system modeling to be integrated into the curriculum.

5. FAQs

Will Splash models follow the XMILE standard for SD models?

The XMILE (XML Modeling Interchange LanguagE) standard describes a standard format for sharing and distributing System Dynamics models[3]. Unlike existing system dynamics software for which the XMILE standard was developed, Splash will make extensive use of a spatial physics simulation in all its models. Because of this fundamental difference, Splash will not be able to make use of the XMILE standard. Later versions of the design may seek to improve Splash's compatibility with the XMILE standard.

How will units be specified in Splash?

Units can be added or changed via the object's Property Panel (see Figure 5.1) by typing in any unit name you desire. Once a unit is specified in a model, the same unit can then be re-used for other containers and pipes. Units of stocks and flows will be represented by the liquid material flowing through the system. Different types of liquids in a model will have different colors.

How will unit consistency be enforced?

Given that units are represented by liquids in Splash, unit consistency will be achieved by ensuring that different types of liquids do not mix. Containers will check for unit consistency by only blocking and collecting the liquid for which they are intended. For example, if the unit for a container is 'Dollars' and a flow of 'People' falls into that container, the flow will simply pass through the container, unobstructed. Where possible, units will be automatically assigned.

[3] For more information on XMILE visit xmile.systemdynamics.org

Fig 5.1: Property Panel for a container

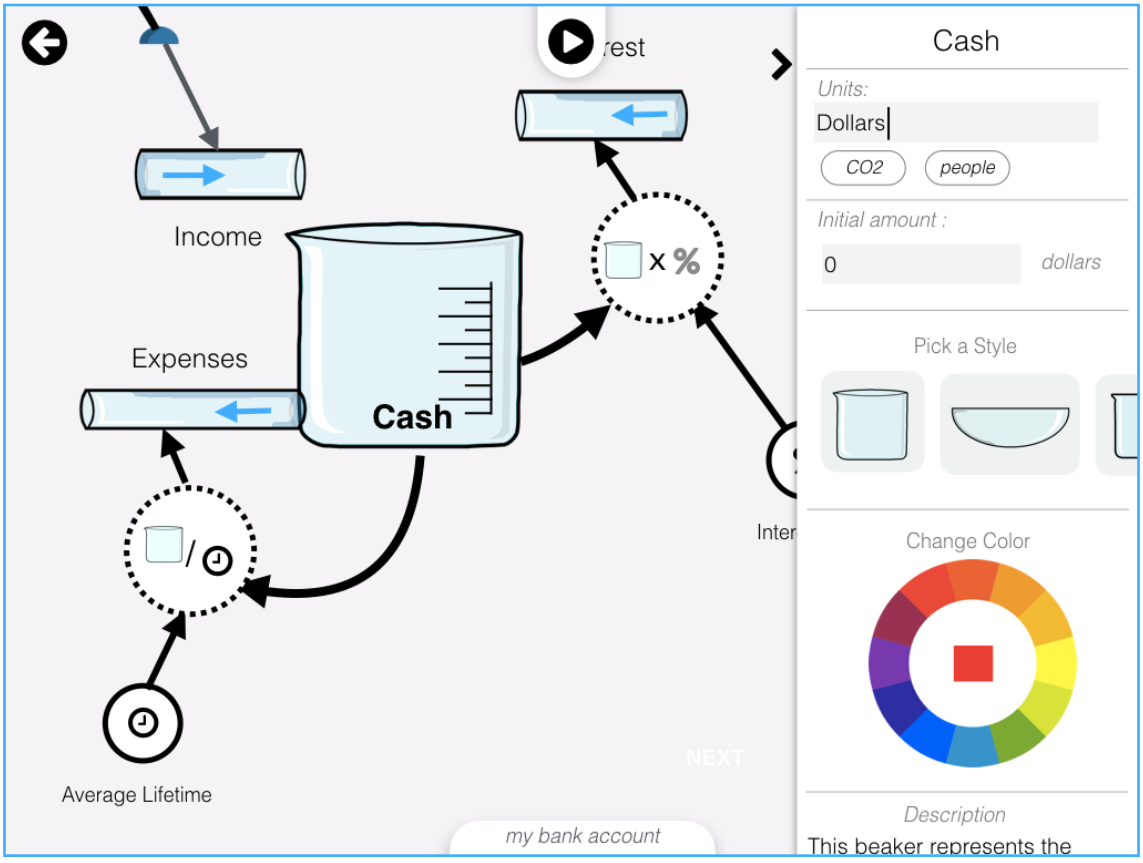
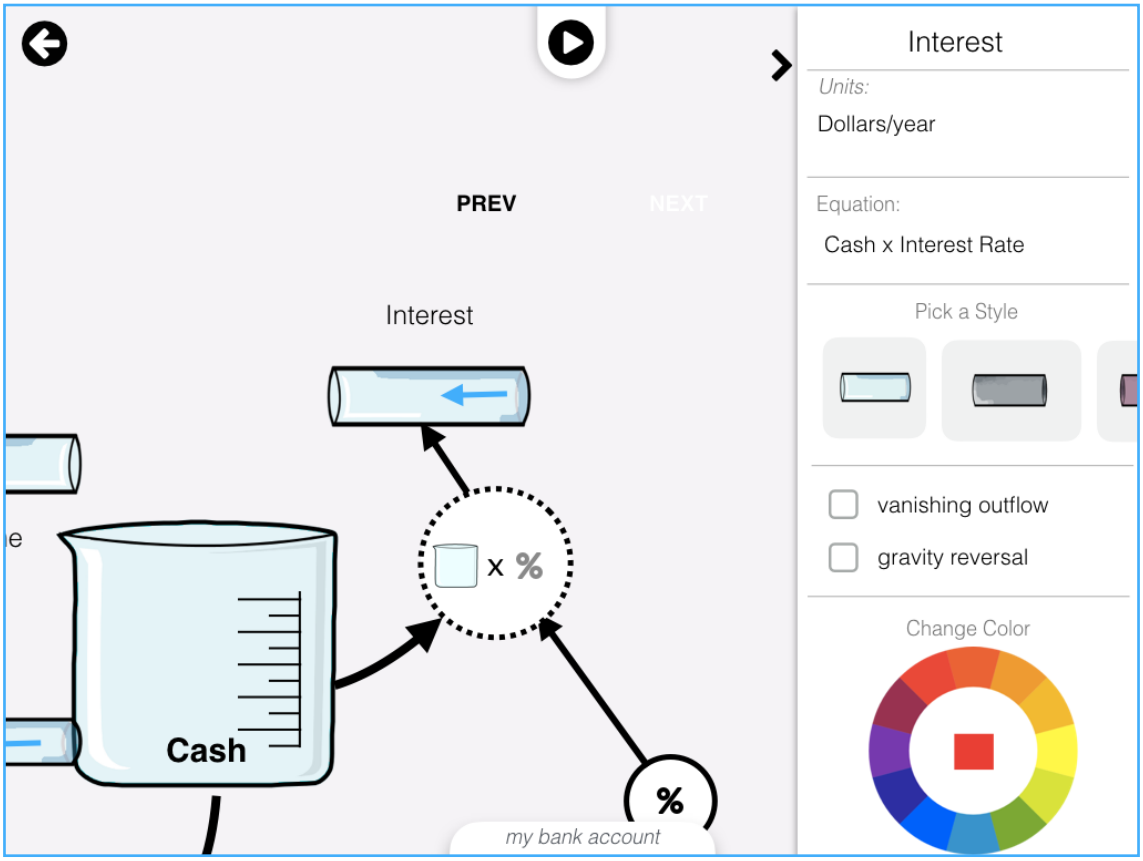


Figure 5.2: Property Panel for a pipe



How will I change the color of the liquid?

To change the color of a liquid, you'll open up the Property Panel of any associated container or pipe and then use the color wheel provided to pick any shade and tone you like (refer figure 5.1, 5.2).

Are the size and shape of containers and pipes fixed?

Via the Property Panel you will have the option to choose from a limited selection of container shapes (see figures 5.1). Similarly, you'll also be able to change the visual appearance of a pipe. Once a container or pipe is selected, a rectangular bounding box will appear. Dragging on the boundary handles will resize it and accordingly adjust the width and height of the object.

If a container is filled beyond its capacity, will the excess liquid overflow?

Yes. Considering that Splash employs a physics simulation, if the net volume of liquid flowing into a container exceeds the container's capacity, it will result in an overflow. This overflow amount will not show up in the graphs as those will only track how much liquid is inside a given container.

Will Splash support arrays?

No.

Can stocks in a Splash model have negative values?

No.

Is there any way to specify mathematical equations in Splash without using the visual math operator elements?

No.

Will Splash have storytelling?

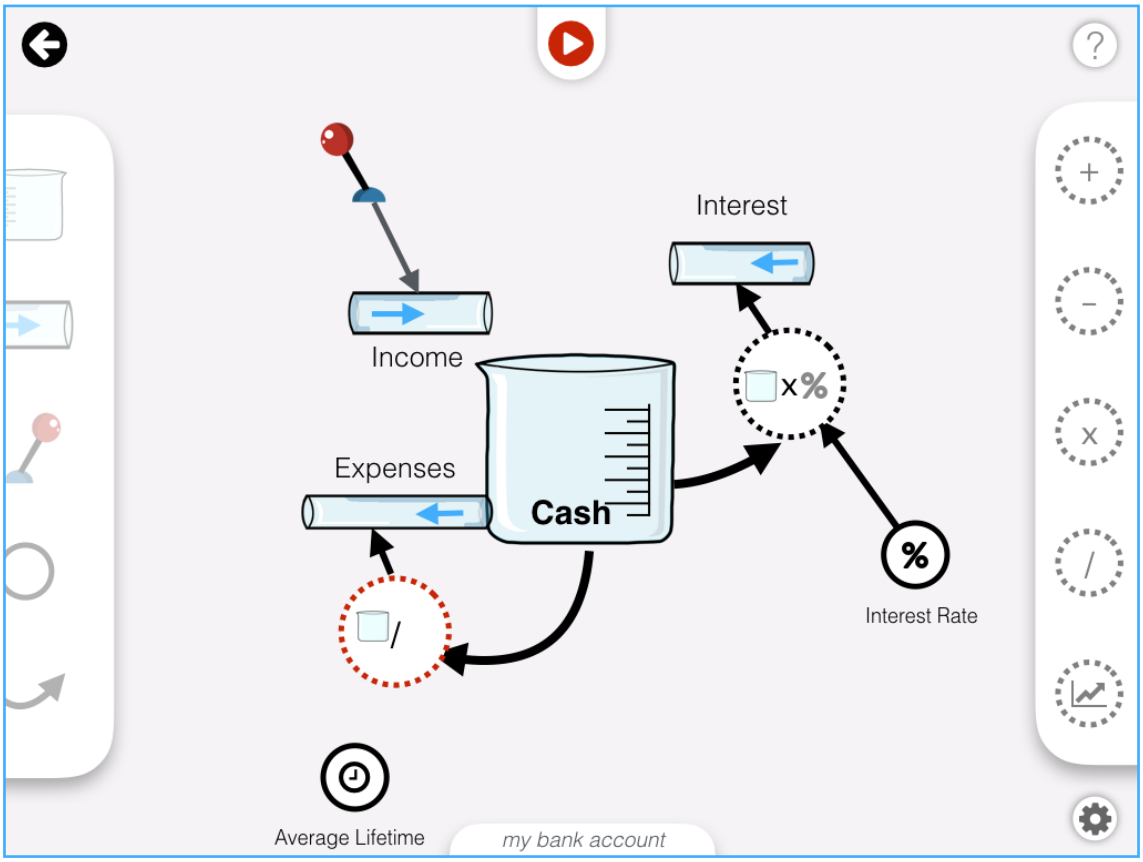
Storytelling in an important part of effectively sharing and communicating complex models. However, the initial version of Splash will not have storytelling functionality. Later iterations of the design may address this.

Will a Splash model always be ready to simulate?

All model elements in Splash come with default values. Because of this, Splash models will nearly always be ready to simulate. The only case in which a model cannot be

simulated is if there is an incomplete mathematical operation. In this case, pressing the play button will not simulate the model. Instead, the play button will turn red to show that something is wrong and the problem element will be highlighted for the user to fix (refer Figure 5.3).

Fig. 5.3: Highlighting incomplete equations



What sort of learning scaffolding will Splash have?

Apart from having a tour of the UI and a contextual help button on most screens, Splash will come with a set of sample models built-in. These sample models will be aimed at helping users learn the concepts of system dynamics and guiding them through the process of effectively using Splash.

On what devices and platforms will Splash be available?

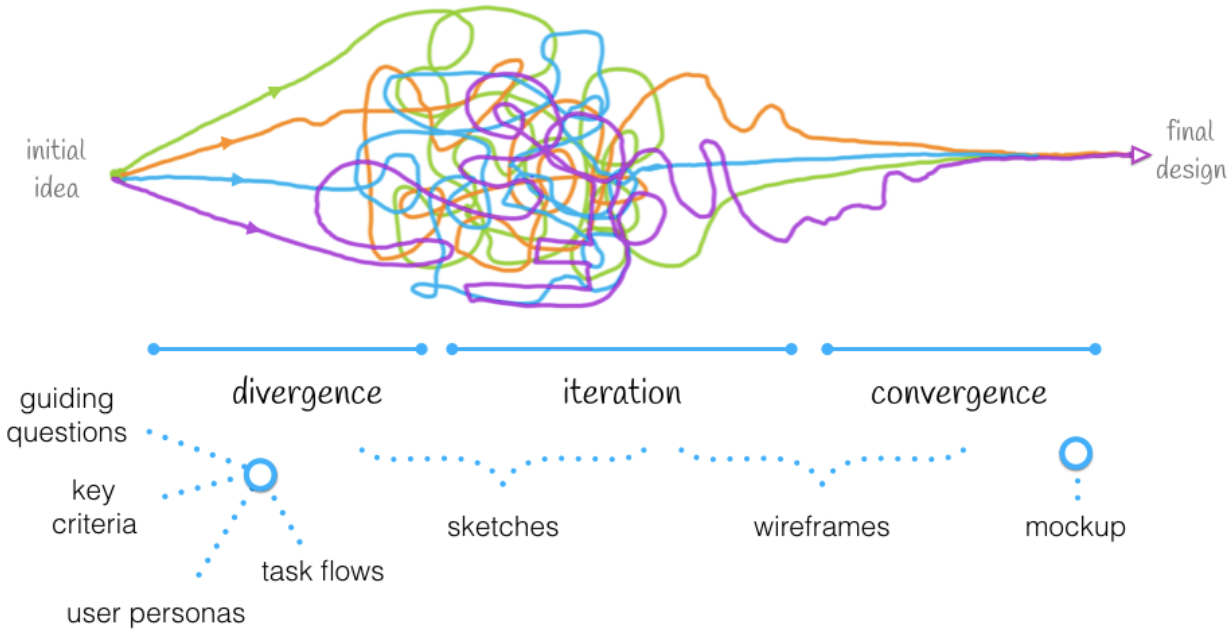
Splash is primarily designed for tablets and mobile phones. It will be developed for iOS and Android devices. A modified version of the app will also be made available for download on Chromebooks. Later iterations of the software may support even more device form factors and operating systems.

Behind the Scenes

A. Our Design Process

Figure A.1 illustrates the generic nature of any design process. It often begins with a problem that needs to be solved or an idea that you want to turn into reality. It's exciting and full of possibility. But as you dive deeper into this idea and accommodate different points of view, you're likely to find that it's more complex and chaotic than initially expected. At this point in the process, it's highly likely that you end up with a convoluted design that superficially addresses everyone's individual concerns but misses the mark overall. The biggest challenge is finding your way through this forest of chaos in a manner that reveals a design that is both effective and fundamentally elegant.

Figure A.1: The Design Process



To improve our chances at revealing such a design, we put together a design framework that would help us navigate when times got tough. This framework consisted of *Guiding Questions, Key Design Criteria, User Personas* and a set of *Use Cases.* Details of the design framework are presented in the following section.

Once the framework was sufficiently identified, initial design candidates were sketched out for discussion and debate. These design candidates were evaluated based on the Guiding Questions and Key Criteria. Design elements that seemed to perform well were adopted and merged, while others were archived. In this Darwinian manner, the design candidates evolved into a single better design. As this final design emerged, its visual representations

moved from low-fidelity sketches and wireframes to higher fidelity wireframes and a clickable prototype.

B. Our Design Framework

B.1. Guiding Questions

Guiding Questions are a select set of questions that help outline the fundamental purpose of the design effort. These questions were intended to help us quickly sift through design ideas at a high level to find those that fit our objective and reject the rest. The Guiding Questions behind the design of Splash are:

a) What will help a beginner learn SD?

This question was further clarified as:

i. *What will guide a beginner from a blank sheet to successfully simulating SD models with minimum of expert support?*
ii. *What will make an SD modeling software self-evident?*

b) What is the best way to honor and communicate core SD principles?
c) What will make SD more intuitively appealing?

B.2. Key Design Criteria

Design Criteria describe the various aspects along which the design must perform well. They collectively serve as a multi-dimensional yardstick that can be used to evaluate and compare design options.

The Key Design Criteria for Splash include:

a) True to SD: *The extent to which core SD principles are honored*

i. *Accumulation*
ii. *Endogeneity*
iii. *Causality*
iv. *Aggregation*

b) Ease of Use

i. *Self-evident design*
ii. *Teacher-friendly*

c) Engagement

i. *Tells a story*
ii. *Supports curiosity*

d) Development Effort: *Time and effort involved in building the design*

e) Shareability: *Software being adopted as a standard tool*

f) Accessibility: *Compatible with available computing devices*

i. *Laptops/Desktops*
ii. *Mobile Phones*
iii. *Tablets*

g) Flexibility: Allows the user to make a wide variety of models

h) Relatability: More concrete than abstract

B.3. User Personas

User Personas are descriptions of potential users of Splash. Personas are intended to capture the relevant goals and behaviors of people in the target audience. They help focus and humanize the numerous design decisions and ensure that the conversation stays relevant to the intended users.

In more extensive design efforts, user personas are developed after a significant amount of user interviews, market research and field observations. However, due to time and resources considerations, the user personas for Splash were developed based on the group's extensive experience interacting with the pre-university teachers and students who form our target audience.

A total of five user personas were identified. Three of them represent pre-university students and two represent pre-university teachers. They are:

The Students:

Ellie
Age: 10
Sex: Female

- High logical/mathematical intelligence
- Has a natural talent for music
- Finds math and science fun
- Does not care much about conforming to norms
- Hates things that slow her down

Jamal
Age: 12
Sex: Male

- Interested in systems thinking because of its potential for interesting insights
- Finds math and equations boring
- Enjoys working with others; Not a fan of lecture style teaching
- Wants to make a significant contribution to the world

Maya
Age: 14
Sex: Female

- Loves painting; Has worked hard to become skilled at it
- Comfortable with math
- Disturbed by poor aesthetics
- Understands the importance of practice
- Understands the value of respecting established norms

The Teachers

Mary
Age: 32
Sex: Female

- Comfortable with ambiguity
- Largely endogenous perspectives
- Has a growth mindset
- Believes she can learn a lot from her students
- Believes that students are capable of exceeding expectations, including their own

Abe
Age: 50
Sex: Male

- Not comfortable with ambiguity
- Largely exogenous perspectives
- Knows a lot (he thinks so)
- Underestimates students' potential
- Not a co-learner

B.4. Use Cases

Use Cases are a set of steps that a user might perform while working with the software to achieve a particular objective. Two use cases were developed to aid the design of Splash. The first follows a user building a new model from scratch while the second is about a user modifying an existing model. Together, these use cases cover a large percentage of the core functionality of the software.

Use Case A:

i. Start a new model
ii. Create the model structure
iii. Specify equations and other properties
iv. Adjust simulation settings
v. Simulate the model
vi. Make changes to the model and re-simulate

Use Case B:

i. Import an existing model
ii. Duplicate the model to have a copy of the original
iii. Run the model and save the run as a reference
iv. Modify the model structure
v. Re-simulate the model and compare runs
vi. Export the modified model

C. Evolution of the Design

C.1. Initial Design

Based on the intent captured in the design framework, an initial design candidate was conceptualized. Figures C.1.1 - C.1.3 show some of the wireframes developed to represent this initial design. The design adopted the common scheme for visualizing system dynamics models – that of using boxes to represent stocks, thick arrows to represent flows, and thin arrows to show connections (refer Figure C.1.1).

In order to reduce the visual clutter that is common in system dynamics models, it would be possible to create the model structure at various levels of 'depth'. The structure at deeper levels would appear translucent and only become completely visible on zooming into the model. On simulating the model, the images for the stocks, flows and auxiliaries would grow and shrink to visually highlight the changes in magnitude over time (Figure C. 1.2).

In addition to this, the behavior of all variables over time would be automatically plotted in a set of graphs. The model structure could be color-coded by the user so as to visually emphasize the relationship between a stock and its flows (Figure C.1.3). Any such color coding would continue into the graphs to make the dynamics easier to spot.

This initial design performed well along several design criteria (such as *True to SD, Development Effort, Accessibility).* A lot of design features conceived as part of this initial design have made their way into Splash. For example, the creation toolbar on the side of the canvas, the model edit/play modality, the side Property Panel, and the auto-generated graph panel can all be seen in the final design.

However, the initial design was not perceived as being sufficiently engaging or extremely easy to use. For example, the play/edit modes accessed via the toggle button on the bottom of the screen could have potentially confused and frustrated new users. The 'box-and-arrow' visualization scheme did not seem like it would be very engaging for beginning learners such as pre-university students. The similarity of this visual scheme to other SD modeling tools that are currently available in the market also raised valid questions about whether this design effort was worthwhile.

Fig. C.1.1: Initial Design: Edit mode

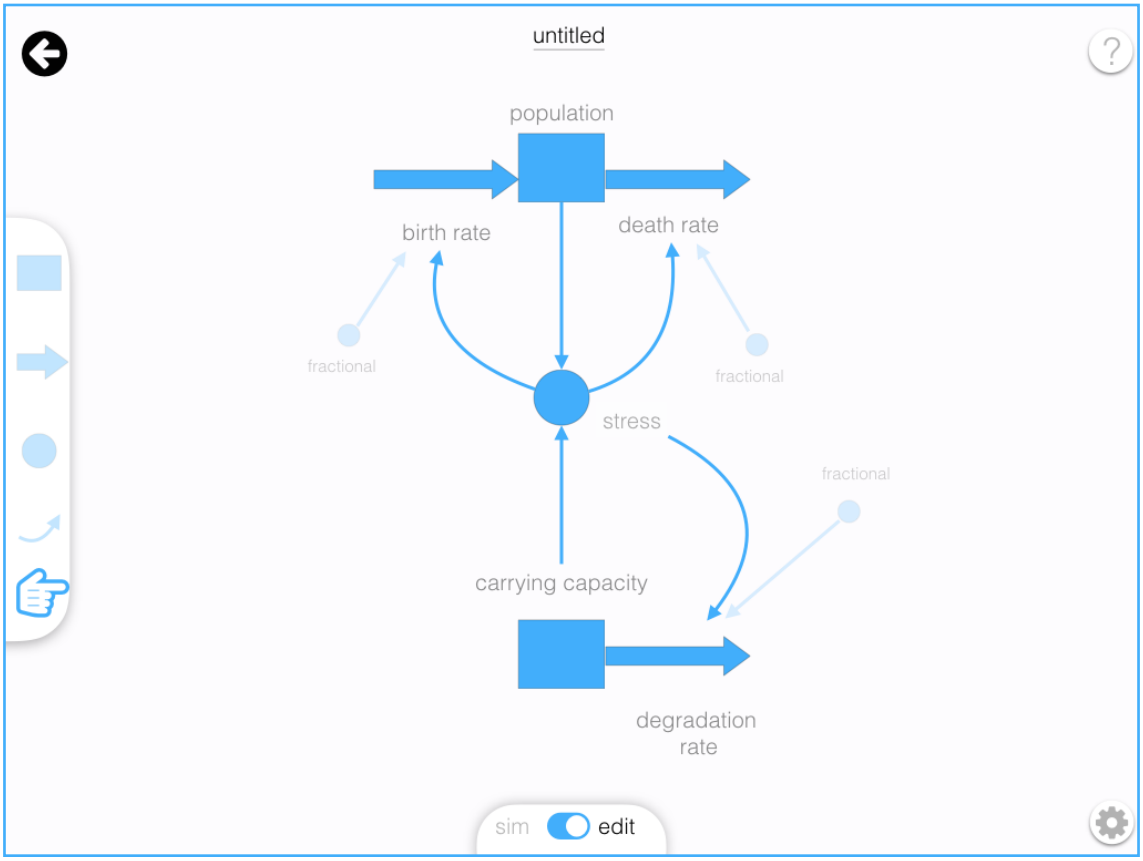


Fig. C.1.2: Initial Design: Model simulation

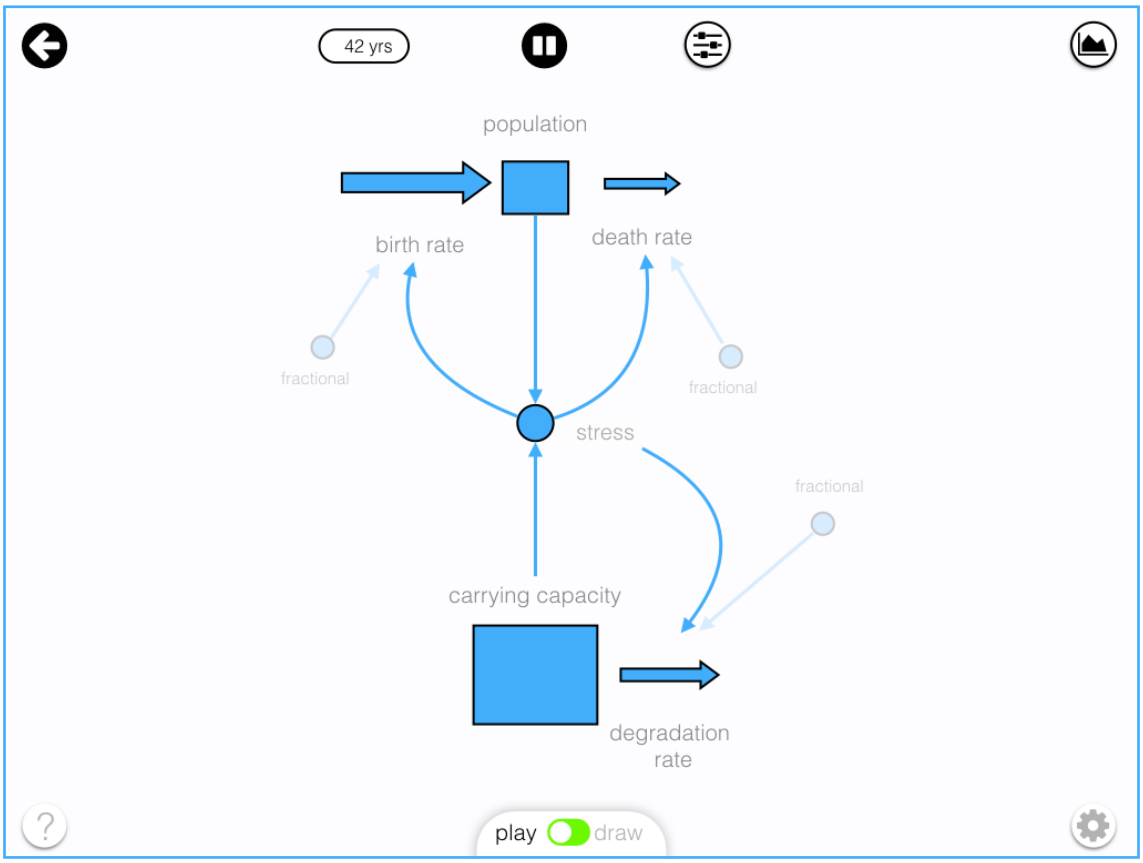
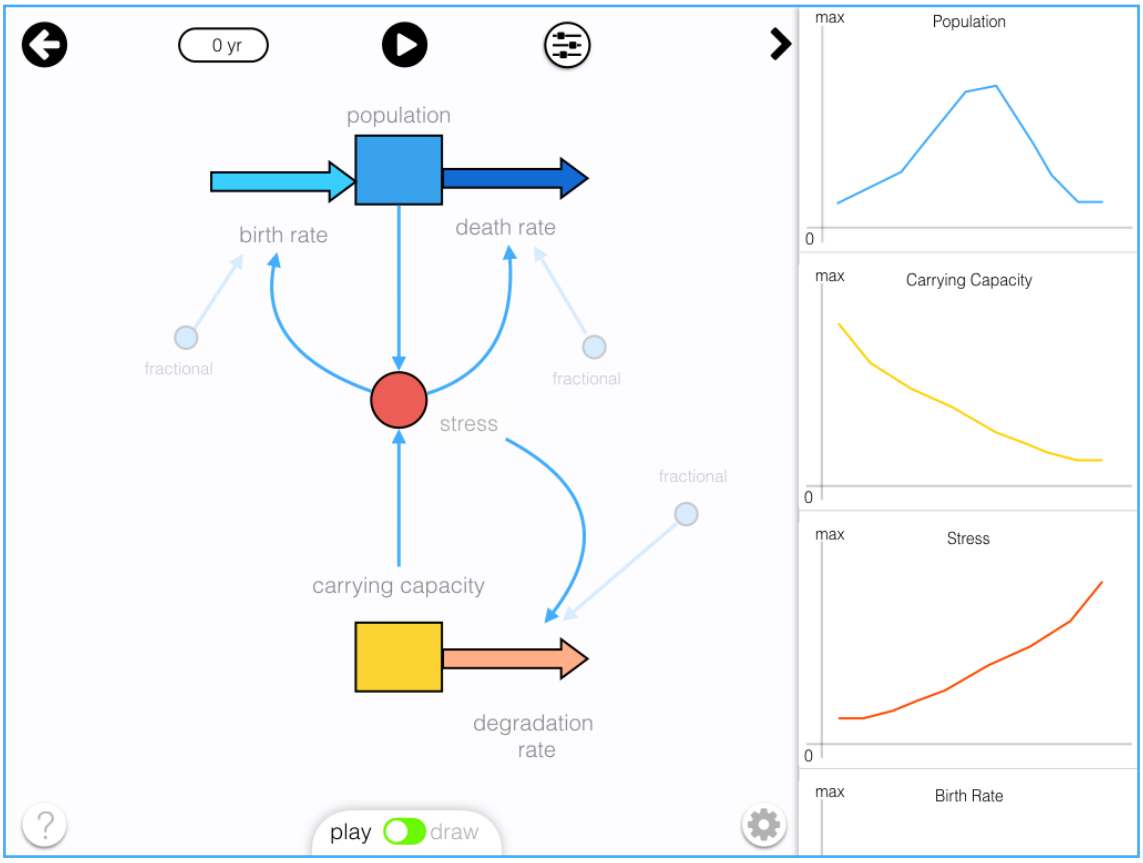


Fig. C.1.3: Initial Design: Graphs and color coding

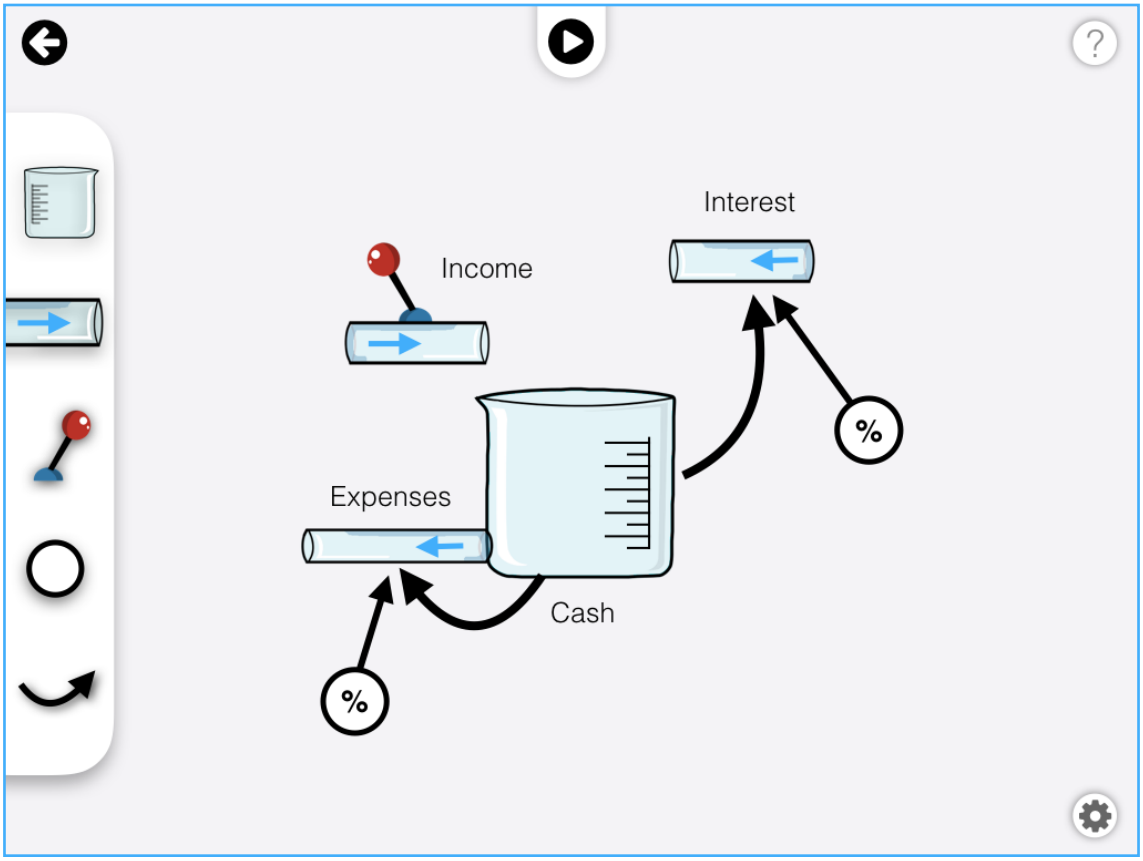


C.2. Using physics

One of the ways in which people have tried to make system dynamics easier and more fun to learn is by using tubs, pipes and pumps to create models in real, physical space. Learners can use such setups to simulate the dynamics of the model by pumping water from one tub to another through the pipes.

We considered that if the key aspects of such kinesthetic exercises were combined with our initial design, it would likely boost the design's engagement factor. Accordingly, we pivoted away from our initial 'box-and-arrow' approach and worked to combine liquid physics simulation with system dynamics in a single design. Figure C.2.1 shows the wireframe for the bank account model from this phase.

Fig. C.2.1: Liquid Physics Concept: No visual math



C.3. Adding Visual Math

Notice that even after the pivot to a physics simulation, the revised design showed in figure C.2.1 has no toolbar for mathematical operators. At this stage, we had imagined that users would first add the model elements onto the canvas and then use the Property Panel to specify their equations. This is a workable approach and is quite common in existing SD software. But it seemed like the most the fun part of SD modeling is simulating the model and watching the dynamics unfold. Reducing the number of steps involved in going from a blank canvas to a model that can be simulated would let users get to the fun stuff faster and potentially make the learning process more appealing.

Our initial concept for making the models simulation-ready was to have the software guess the equations based on the inputs. The challenge with this approach is that arriving at the 'correct' equation for the model structure will require the software to have extensive domain knowledge and some form of artificial intelligence. Guessing equations without any domain knowledge could result in the software getting them wrong. In this case, users would probably have to scan through every equation in the model to find the one(s) they need to adjust. In effect, trying to guess the equations might have needed just as many, if not more, steps for a user to go from a blank canvas to simulating a model.

As a middle-path, we chose to give every stock, flow and auxiliary created a default value and then let users visually create the math needed in the model using operator elements (Figure C.3.1). Having operator elements enables users to conveniently add basic arithmetic equations and graphical inputs into the model without needing to use the Property Panel. Further, by limiting each operator element to only two inputs we entirely avoided the need to guess equations. Equations with more than two variables may still be constructed by connecting the results from multiple operator elements in series. A side benefit of including operator elements is that they boost the transparency of the models, making them less of a black-box.

Fig. C.3.1: Adding visual math