

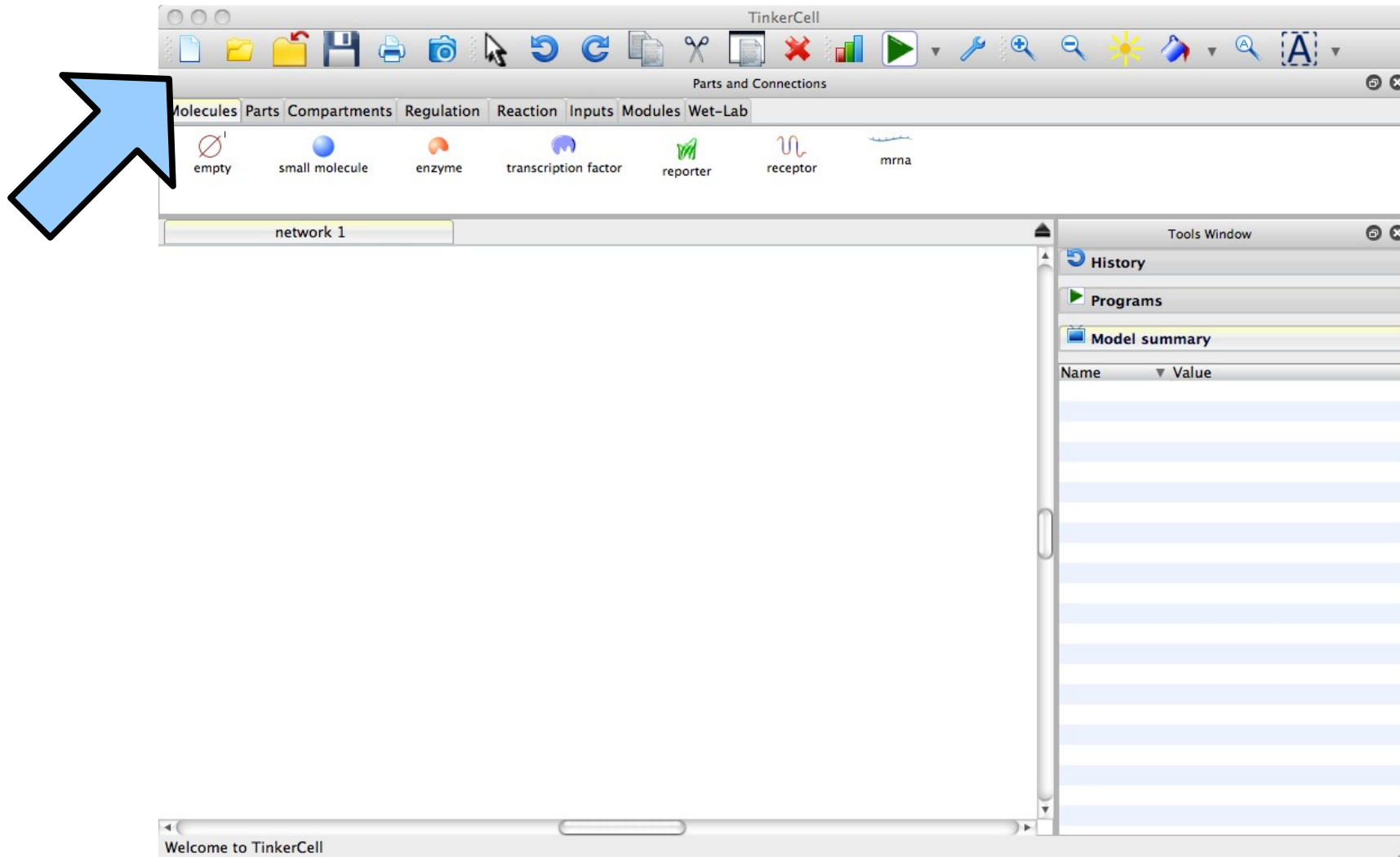
cell

TinkerCell Tutorial: The Bacterial Photography System

Thanks and appreciation to Deepak Chandra from U of W, who developed TinkerCell and modified it to teach the bacterial photography system. The system itself was made by the 2006 Univ. Texas iGEM team.

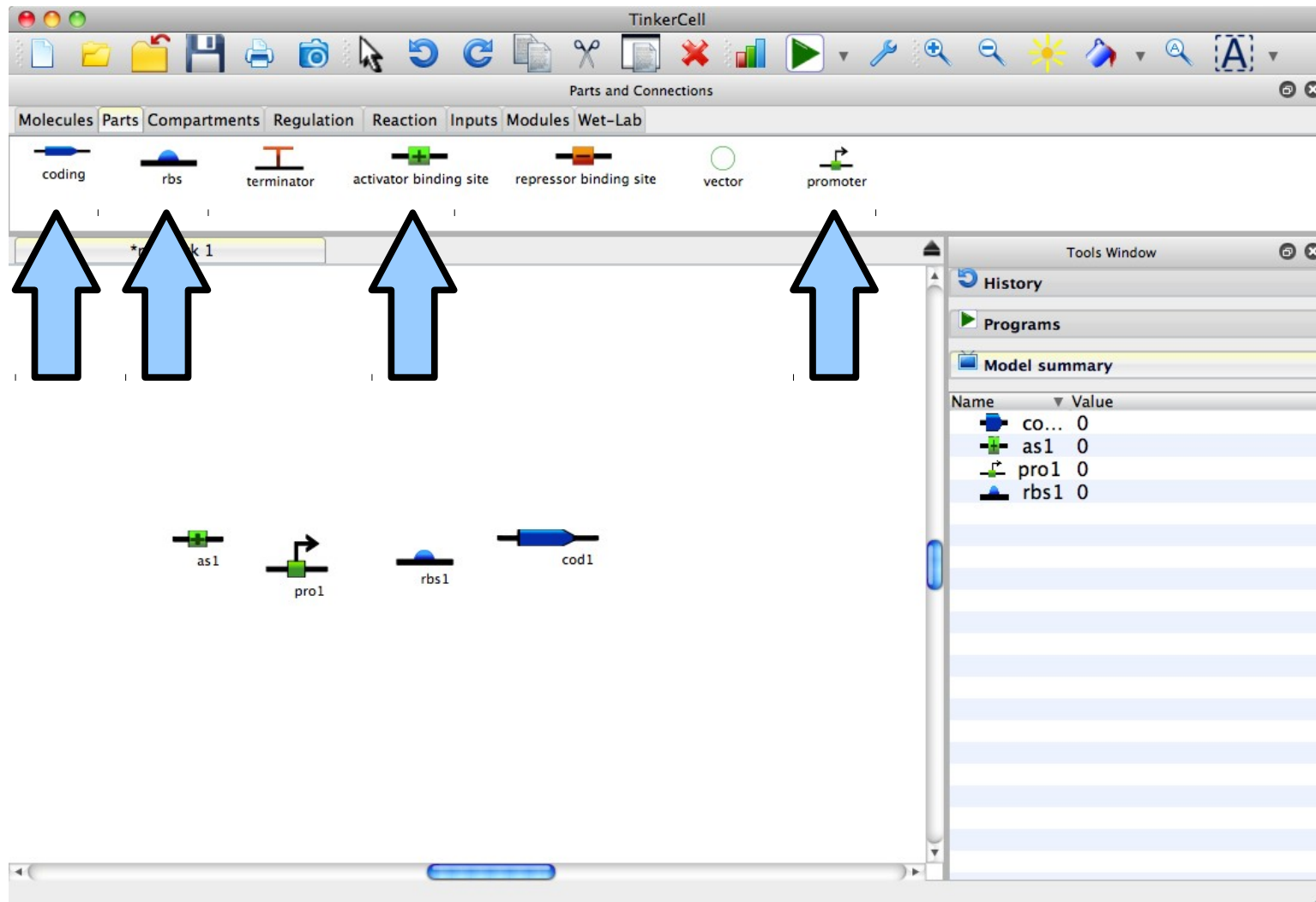
Start a new canvas

Select "New canvas" from the File menu or use the "New" icon at the left of the top toolbar



Assemble Reporter

From the “Parts” tab, select and place an “Activator Binding Site”, “Promoter”, “RBS”, and “Coding” icon on the canvas.



Align and Name

Drag items on the canvas next to one another to align them. Rename the promoter “PompC” and the coding sequence “LacZ”.

The screenshot shows the TinkerCell software interface. The top toolbar contains various icons for file operations, editing, and simulation. Below the toolbar is the 'Parts and Connections' panel with tabs for 'Molecules', 'Parts', 'Compartments', 'Regulation', 'Reaction', 'Inputs', 'Modules', and 'Wet-Lab'. The 'Parts' tab is active, showing icons for coding, rbs, terminator, activator binding site, repressor binding site, vector, and promoter.

The main canvas displays a genetic circuit diagram with the following components from left to right: an activator binding site (as1), a promoter (PompC), a ribosome binding site (rbs1), and a coding sequence (LacZ). The promoter is represented by a green square with a right-pointing arrow, and the coding sequence is a blue arrow pointing right.

On the right side, the 'Tools Window' is open, showing a 'Model summary' table with the following data:

Name	Value
La...	PompC.strength * (as1)
as1	0
Po...	0
rbs1	0

At the bottom left of the interface, a status bar indicates 'items moved by (-6,16)'.

Add a transcription factor

From the “Molecules” tab, add a transcription factor to the canvas and rename it OmpRp

The screenshot shows the TinkerCell software interface. The 'Molecules' tab is selected, and the 'transcription factor' icon is highlighted with a blue arrow. The canvas displays a genetic circuit with components 'as1', 'PompC', 'rbs1', and 'LacZ'. A 'Tools Window' on the right shows a 'Model summary' table with the following data:

Name	Value
O...	0
La...	PompC.strength * (as1)
as1	0
Po...	0
rbs1	0

tf1 renamed to OmpRp

Visual appeal: add a 'P'

Select OmpRp, then from the “Edit” menu, choose “Add Decorator”. From the “Decorator” tab, select “phosphorylation”.

The screenshot displays the TinkerCell software interface. The main workspace shows a genetic circuit diagram with components: as1 (green square), PompC (green square with arrow), rbs1 (blue circle), and LacZ (blue arrow). A legend above the diagram shows OmpRp (blue shape with a red 'P' in a yellow circle). The 'Parts and Connections' panel at the top lists various biological parts like empty, small molecule, enzyme, transcription factor, reporter, receptor, and mrna. The 'Tools Window' on the right contains a 'Model summary' table with the following data:

Name	Value
O...	0
La...	PompC.strength * (as1)
as1	0
Po...	0
rbs1	0

At the bottom left, a status bar indicates 'items moved by (-20,-24)'.

Regulate transcription

From the “Regulation” tab, choose “transcription regulation”, then click OmpRp and as1. Choose “transcriptional activation” from the pop-up.

The screenshot shows the TinkerCell software interface. The 'Regulation' tab is selected, and the 'transcription regulation' icon is highlighted with a blue arrow. The diagram below shows a genetic circuit where OmpRp (a blue protein) binds to the promoter of the as1 gene (a green box), activating its transcription. The as1 gene produces a green protein that binds to the promoter of the LacZ gene (a blue box), activating its transcription. Other components include PomC (a green box) and rbs1 (a blue box).

The Tools Window on the right shows the following model summary:

Name	Value
ta1	<grouped>
O...	0
La...	PompC.strength * (as1)
as1	$((1+((OmpRp/ta1.Kd)^{n_{act}}))^{-1})$
Po...	0
rbs1	0

The Console Window at the bottom right shows the following output:

```
>Failed to initialize Ruby
>Ruby not loaded
>TinkerCell Python module imported
Numpy imported
Scipy not loaded
Pysces not loaded
NetworkX not loaded
>1.0 does not evaluate
```

The status bar at the bottom left shows "ta1 inserted".

Add light receptor

From the “Molecules” tab, place a “Receptor” on the canvas and rename it Cph8.

The screenshot displays the TinkerCell software interface. The top toolbar contains various icons for file operations and simulation. Below the toolbar is the 'Parts and Connections' panel, which is currently set to the 'Molecules' tab. This panel shows several molecular components: empty, small molecule, enzyme, transcription factor, reporter, receptor, and mrna. A large blue arrow points to the 'receptor' icon. Below this panel is a canvas area where a genetic circuit is being constructed. The circuit includes a transcription factor 'OmpRp' that regulates the expression of 'as1'. 'as1' in turn regulates 'PompC', which is followed by 'rbs1' and 'LacZ'. A 'Cph8' receptor icon is also present on the canvas. On the right side, there is a 'Tools Window' with a 'Model summary' table and a 'Console Window' showing the execution log.

Name	Value
ta1	<grouped>
O...	0
Cp...	0
La...	PompC.strength * (as1)
as1	$((1+((OmpRp/ta1.Kd)^{n...}))^{-1})$
Po...	0
rbs1	0

```
>TinkerCell Python module imported
Numpy imported
Scipy not loaded
Pysces not loaded
NetworkX not loaded
>1.0 does not evaluate
```


Regulate OmpRp with Cph8 with Cph8

From the “Regulation” tab, choose “Allosteric Inhibition” and then click on Cph8 and OmpRp.

The screenshot displays the TinkerCell software interface. The 'Regulation' tab is active, and the 'allosteric inhibition' icon is highlighted with a blue arrow. The main workspace shows a genetic circuit diagram with the following components and connections:

- Cph8 (wavy line) is connected to ai1 (red circle) via a red arrow.
- ai1 is connected to OmpRp (blue protein) via a red arrow.
- OmpRp is connected to ta1 (green circle) via a green arrow.
- ta1 is connected to as1 (green square) via a green arrow.
- as1 is connected to PompC (green square) via a green arrow.
- PompC is connected to rbs1 (blue circle) via a blue arrow.
- rbs1 is connected to LacZ (blue arrow) via a blue arrow.

The right sidebar contains a 'Tools Window' with a 'Model summary' table and a 'Console Window' with execution logs.

Name	Value
ai1	<grouped>
ta1	<grouped>
O...	0
Cp...	0
La...	PompC.strength * (as1)
as1	$((1+((OmpRp/ta1.Kd)^t...$
Po...	0
rbs1	0

```
>TinkerCell Python module imported
Numpy imported
Scipy not loaded
Pysces not loaded
NetworkX not loaded
>1.0 does not evaluate
```

ai1 inserted

Add the BetaGal protein

From the “Molecules” tab, place an “Enzyme” on the canvas. Name it “BetaGal”.

The screenshot shows the TinkerCell software interface. The 'Molecules' tab is selected, and the 'enzyme' icon is highlighted with a blue arrow. The main canvas displays a genetic circuit diagram with components like Cph8, ai1, OmpRp, ta1, as1, PompC, rbs1, and LacZ. A 'BetaGal' enzyme icon is also present. The right sidebar shows a 'Tools Window' with a 'Model summary' table and a 'Console Window' with execution logs.

Name	Value
ai1	<grouped>
ta1	<grouped>
Bet...	0
O...	0
Cp...	0
La...	PompC.strength * (as1)
as1	((1+((OmpRp/ta1.Kd)...
Po...	0
rbs1	0

```
>TinkerCell Python module imported
Numpy imported
Scipy not loaded
Pysces not loaded
NetworkX not loaded
>1.0 does not evaluate
```

enz1 renamed to BetaGal

Produce BetaGal protein

From the “Reactions” tab, choose “Protein Production”. Link the LacZ coding region to the BetaGal enzyme.

The screenshot displays the TinkerCell software interface. The top toolbar contains various icons for file operations and simulation. Below the toolbar, the 'Parts and Connections' panel is active, showing the 'Reaction' tab. The 'Reaction' tab includes icons for different reaction types: activation, repression, 1 to 1, 2 to 1, 3 to 1, 2 to 2, 3 to 2, 2 to 3, 3 to 3, protein production, and induced transcript. A blue arrow points to the 'protein production' icon. The central workspace shows a genetic circuit diagram. The circuit includes a promoter (Cph8) driving an activator (ai1), which in turn activates a repressor (OmpRp). OmpRp represses a promoter (ta1), which drives the expression of as1. as1 activates a promoter (PompC), which drives the expression of rbs1. rbs1 represses the LacZ coding region, which is linked to the BetaGal enzyme. The right sidebar shows the 'Model summary' table and the 'Console Window'.

Name	Value
pp1	<grouped>
ai1	<grouped>
ta1	<grouped>
Bet...	0
O...	0
Cp...	0
La...	PompC.strength * (as1)
as1	((1+((OmpRp/ta1.Kd)...))
Po...	0

```
>TinkerCell Python module imported
Numpy imported
Scipy not loaded
Pysces not loaded
NetworkX not loaded
>1.0 does not evaluate
```

Add the colorful molecule

From the “Molecules” tab, place two “small molecules” on the canvas. Name the first “SGal” and the second “COLOR”.

The screenshot displays the TinkerCell software interface. The top toolbar includes icons for file operations, navigation, and simulation. Below the toolbar is the 'Parts and Connections' panel with tabs for 'Molecules', 'Parts', 'Compartments', 'Regulation', 'Reaction', 'Inputs', 'Modules', and 'Wet-Lab'. The 'Molecules' tab is active, showing icons for 'empty', 'small molecule', 'enzyme', 'transcription factor', 'reporter', 'receptor', and 'mrna'. A blue arrow points to the 'small molecule' icon. The main canvas shows a genetic circuit diagram with components: Cph8 (red wavy line), ai1 (blue square), OmpRp (blue protein), ta1 (green square), as1 (green square), PompC (green arrow), rbs1 (blue circle), LacZ (blue arrow), and BetaGal (orange protein). Two small molecule icons, labeled 'SGal' and 'COLOR', are placed on the canvas. The 'Tools Window' on the right shows a 'Model summary' table with the following data:

Name	Value
Bet...	0
O...	0
CO...	0
SGal	0
ai1	<grouped>
Cp...	0
La...	$PompC.strength * (as1)$
as1	$((1+((OmpRp/ta1.Kd)...))$
Po...	0

The 'Console Window' at the bottom right shows the following output:

```
>TinkerCell Python module imported
Numpy imported
Scipy not loaded
Pysces not loaded
NetworkX not loaded
>1.0 does not evaluate
```

The status bar at the bottom left indicates 'sm2 renamed to COLOR'.

Supply SGal

Double-click on SGal, select the “Initial Conditions” tab, and change its initial concentration from 0 to 15 μM .

The screenshot shows the TinkerCell interface with a network diagram and an open 'Model Summary' dialog box. The network diagram includes components Cph8, ai1, OmpRp, and as1. The 'Model Summary' dialog box is open to the 'Initial Values' tab, showing a table of initial values for selected items. A blue arrow points to the '15 uM' value in the table.

name	value	fixed?
SGal	15 μM	floating

changed initial values or fixed

Add the color reaction

From the “Regulation” tab, choose “Enzyme Catalysis”, then click on BetaGal, SGal, and COLOR.

The screenshot displays the TinkerCell software interface. The top menu bar includes 'Molecules', 'Parts', 'Compartments', 'Regulation', 'Reaction', 'Inputs', 'Modules', and 'Wet-Lab'. The 'Regulation' tab is active, showing icons for 'enzyme catalysis', 'allosteric inhibition', 'transcription regulation', and 'direct gene regulation'. A blue arrow points to the 'enzyme catalysis' icon. Below the menu is a workspace labeled '*network 1' containing a genetic circuit diagram. The diagram shows a DNA strand with several elements: a promoter 'as1', a gene 'PompC', a ribosome binding site 'rbs1', and a gene 'LacZ'. A protein 'OmpRp' is shown binding to the 'as1' promoter, with a red arrow indicating inhibition. A protein 'ai1' is shown binding to the 'PompC' promoter, also with a red arrow indicating inhibition. A protein 'ta1' is shown binding to the 'LacZ' promoter, with a green arrow indicating activation. A protein 'pp1' is shown binding to the 'LacZ' gene, with a green arrow indicating activation. The 'LacZ' gene is transcribed into mRNA, which is then translated into the protein 'BetaGal'. The 'BetaGal' protein is shown catalyzing the reaction of 'SGal' and 'COLOR' into 'ec1'. The 'Tools Window' on the right contains a 'Model summary' table with the following data:

Name	Value
Bet...	0
O...	0
CO...	0
SGal	15
ai1	<grouped>
Cp...	0
La...	PompC.strength * (as1)
as1	((1+((OmpRp/ta1.Kd)...))
Po...	0

The 'Console Window' at the bottom shows the following output:

```
>failed to initialize ruby  
>Ruby not loaded  
  
>TinkerCell Python module imported  
Numpy imported  
Scipy not loaded  
Pysces not loaded  
NetworkX not loaded  
  
>1.0 does not evaluate
```

At the bottom left of the interface, the text 'Rate equations updated' is visible.

Add a chassis

From the “Compartments” tab, choose “Cell”. Place and resize it on the canvas to include all components (but leave Cph8 in the membrane).

The screenshot displays the TinkerCell software interface. At the top, the title bar reads "TinkerCell". Below it is a toolbar with various icons for file operations, editing, and simulation. The main menu bar includes "Molecules", "Parts", "Compartments", "Regulation", "Reaction", "Inputs", "Modules", and "Wet-Lab". The "Compartments" tab is selected, showing a palette with a purple circle labeled "compartment" and a green oval labeled "cell". A large blue arrow points from the "cell" icon down to a green oval on the canvas. The canvas contains a complex biological model with various components: a red wavy line labeled "Cph8" in the membrane; a blue protein labeled "ai1" with a red T-bar; a blue protein labeled "OmpRp" with a red T-bar; a blue protein labeled "ta1" with a green T-bar; a blue protein labeled "SGal" with a blue arrow; a blue protein labeled "ec1" with a blue arrow; a blue protein labeled "COLOR"; a red protein labeled "BetaGal"; a green protein labeled "pp1" with a green arrow; and a DNA segment with a green box labeled "as1", a green box labeled "PompC", a blue box labeled "rbs1", and a blue arrow labeled "LacZ". The green oval is labeled "cell1" at the bottom right. On the right side, there are two windows: "Tools Window" and "Console Window". The "Tools Window" has sections for "History", "Programs", and "Model summary". The "Model summary" section shows a table with "Name" and "Value" columns, containing one entry: "cell1 1". The "Console Window" shows a list of messages in green text: "> failed to initialize ruby", "> Ruby not loaded", "> TinkerCell Python module imported", "Numpy imported", "Scipy not loaded", "Pysces not loaded", "NetworkX not loaded", "> 1.0 does not evaluate", and ">". At the bottom left of the interface, it says "items moved by (8,98)".

Add light

From the “Molecules” tab, place a small molecule outside the cell and rename it “Light”.

The screenshot displays the TinkerCell software interface. At the top, the 'Molecules' tab is selected in the 'Parts and Connections' panel. A blue arrow points to the 'small molecule' icon. Below this, a 'Light' molecule (represented by a blue sphere) is shown being added to the workspace. The main workspace contains a green cell model labeled 'cell1' with various internal components: Cph8, ai1, OmpRp, ta1, as1, PompC, rbs1, LacZ, SGal, ec1, BetaGal, and COLOR. The 'Tools Window' on the right shows a table with the following data:

Name	Value
cell1	1
Light	15

The 'Console Window' at the bottom right shows the following output:

```
> failed to initialize ruby  
> Ruby not loaded  
> TinkerCell Python module imported  
Numpy imported  
Scipy not loaded  
Pysces not loaded  
NetworkX not loaded  
> 1.0 does not evaluate
```

At the bottom left of the interface, a status bar indicates 'sm1 renamed to Light'.

Regulate Cph8 with light

From the "Reaction" tab, choose "Activation" and click on Light and Cph8.

The screenshot shows the TinkerCell software interface. The main window displays a genetic circuit diagram within a cell boundary. The circuit includes a light input (blue sphere) connected to a promoter (aa1) that drives the expression of Cph8 (red wavy line). Cph8 is shown binding to the promoter of the as1 gene, which is part of a larger genetic construct containing PomcC, rbs1, and LacZ. Other components include OmpRp, ta1, SGal, ec1, BetaGal, and COLOR. The interface includes a toolbar with various reaction types, and the 'activation' button is highlighted with a blue arrow. The right-hand side shows a 'Tools Window' with a 'Model summary' table and a 'Console Window' with system messages.

Name	Value
cell1	1
Light	15
aa1	<grouped>

```
>failed to initialize ruby
>Ruby not loaded
>TinkerCell Python module imported
Numpy imported
Scipy not loaded
Pysces not loaded
NetworkX not loaded
>.1.0 does not evaluate
```

aa1 inserted

Turn on the light

From the “Inputs” tab, choose “Step input” and click on the Light molecule.

The screenshot displays the TinkerCell software interface. At the top, the title bar reads "TinkerCell". Below it is a toolbar with various icons for file operations and simulation. The main menu includes "Molecules", "Parts", "Compartments", "Regulation", "Reaction", "Inputs", "Modules", and "Wet-Lab". The "Inputs" tab is selected, showing options for "New event", "Global parameters", "Step input", "Impulse", and "Wave input". A large blue arrow points to the "Step input" option. Below the menu, a network named "*network 1" is visible. The central workspace shows a green cell containing a genetic circuit. A "Light" molecule (blue sphere) is connected to a "Cph8" part (red wavy line), which in turn connects to an "ai1" part (red arrow). The "ai1" part regulates the "OmpRp" part (blue protein), which then regulates the "ta1" part (green arrow). The "ta1" part regulates the "as1" part (green arrow), which is connected to a "PompC" part (green arrow). The "PompC" part regulates the "rbs1" part (blue arrow), which is connected to a "LacZ" part (blue arrow). The "LacZ" part produces a "BetaGal" molecule (red sphere), which is connected to a "pp1" part (green arrow). The "pp1" part regulates the "ec1" part (blue arrow), which produces a "COLOR" molecule (blue sphere). The "SGal" part (blue sphere) is also connected to the "ec1" part. The "Tools Window" on the right shows a "Model summary" table with the following data:

Name	Value
cell1	1
Light	Light.step_height/(1.0 ...
aa1	<grouped>

The "Console Window" at the bottom right shows the following output:

```
> failed to initialize ruby  
> Ruby not loaded  
> TinkerCell Python module imported  
Numpy imported  
Scipy not loaded  
Pysces not loaded  
NetworkX not loaded  
> 1.0 does not evaluate
```

At the bottom left, a status bar indicates "Step function inserted".

Change the timing

In the “Model summary” menu on the right, find “Light” and click the arrow to expand. Change “Step_time” to 50.

The screenshot shows the TinkerCell interface. The main workspace displays a genetic circuit model within a cell boundary. The circuit includes a light input (aa1) that activates a promoter (Cph8), leading to the expression of OmpRp. OmpRp inhibits a promoter (ta1), which in turn inhibits the expression of as1. as1 inhibits the PomC promoter, which drives the expression of rbs1. rbs1 inhibits the LacZ promoter, which produces the LacZ protein. The LacZ protein, along with the SGal protein, forms a complex that inhibits the ec1 promoter, which produces the COLOR protein. The COLOR protein is detected by BetaGal. The model is named '*network 1' and is associated with the cell 'cell1'.

The Tools Window on the right shows the 'Model summary' section. A blue arrow points to the 'Light' entry in the table. The table lists the following parameters:

Name	Value
cell1	1
Light	Light.step_h...
molecularweight	50000
step_height	1
step_time	50
step_steepness	4
aa1	<grouped>

The Console Window at the bottom shows the following output:

```
> failed to initialize ruby  
> Ruby not loaded  
  
> TinkerCell Python module imported  
Numpy imported  
Scipy not loaded  
Pysces not loaded  
NetworkX not loaded  
  
> 1.0 does not evaluate
```

The status bar at the bottom left indicates 'Light.step_time = 50'.