LYNX™

LCSI

CREATING A RACE SIMULATION CODERS' GUIDE

Introduction

Project Goal: Create a computer simulation

In the following six lessons you are going to create a simple racing game, a computer simulation of a sports event, in which you will set the conditions for the racers. Once you have the basics, you can change some of the conditions to see how these changes affect the results.

A computer simulation is a way to represent the real world by coding a computer program that copies the basic features of a real-life situation. Computer simulations are used in all areas of science, from archaeology to zoology and to predict outcomes as diverse as weather patterns, population growth, and even outcomes in sports.

Plan

Before you start, here are some important tips:

- Decide what kind of race you are going to create. Who will be competing? In the sample, there are three people in wheelchairs, but you can use your imagination: the racers can be horses, cars, a turtle and a hare -- anything you can think of (and find the clipart for). In the "All Projects" section of lynxcoding.org, there is a grey folder called "Templates", open it and the Races Template and see some pre-made clipart we have included to help you get started quickly.
- Think of the background setting for your race. Where does it take place? A stadium? A gym? What other features will there be? Are the additional features going to be animated?
- Look at the *On Your Mark, Get Set, GO!* sample project for ideas on how to start your project and see what is possible.
- It is important to remember that Lynx ***does not automatically save your work***, so you must remember to ***save your projects often***!

Here's what you'll do in this project:

- Plan your work, break it into stages. Check, debug and add details.
- Learn the Lynx commands to create turtle movements and animation.
- Add interactive objects, such as buttons and interactive turtles to trigger actions and to restore the starting position.
- Use multiple sprites to create animations.

Introduction

- Add music and/or sound for a true multimedia experience.
- Use and understand “random” events.
- Add event handlers to be used as triggers.
- Learn how to best duplicate objects that behave similarly.
- Add the necessary programming features to set-up and run your simulation multiple times.
- Learn about one of the most important types of data in Lynx – words and use the `word` command to create personalized messages.

Introduction

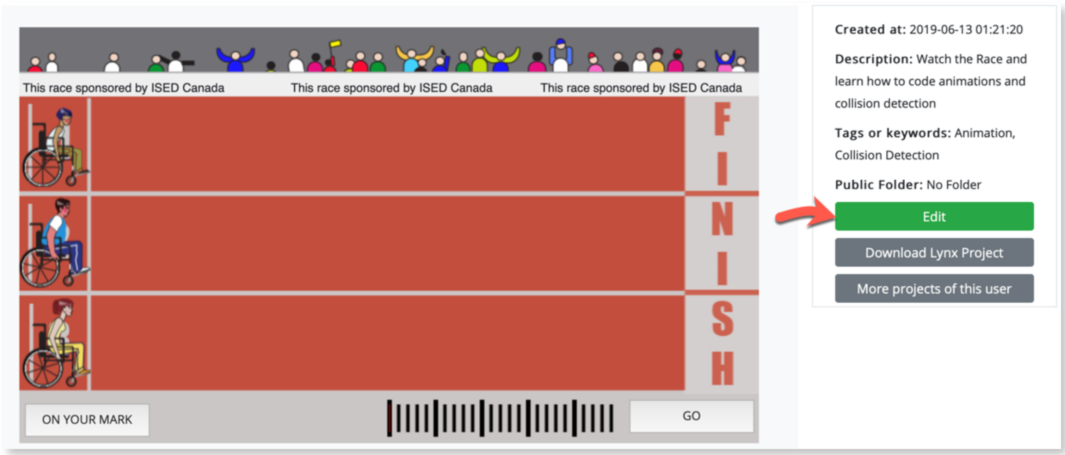
Step 1: Get to know Lynx

You should have already created a Lynx account. (If you haven't, talk to your teacher first).

If this is your first project to be created in Lynx, you may wish to complete Lessons 1-6 in the Story Coders' Guide PDF. It shows you the layout and basic features of Lynx, including the Work area, Command Centre, Procedures Pane and the Clipart Pane. As soon as you understand the layout of Lynx, start your project!

Step 2: Explore the On Your Mark, Get Set, GO! Learner Mode sample

Click on All Projects on the Lynx home page, open the Learner Mode folder, and select the On Your Mark, Get Set, GO! project. The project appears in Play Mode. To see how it was created, log in to Lynx and click the project Edit button.


This race sponsored by ISED Canada
This race sponsored by ISED Canada
This race sponsored by ISED Canada
FINISH
ON YOUR MARK
GO
Edit
Download Lynx Project
More projects of this user

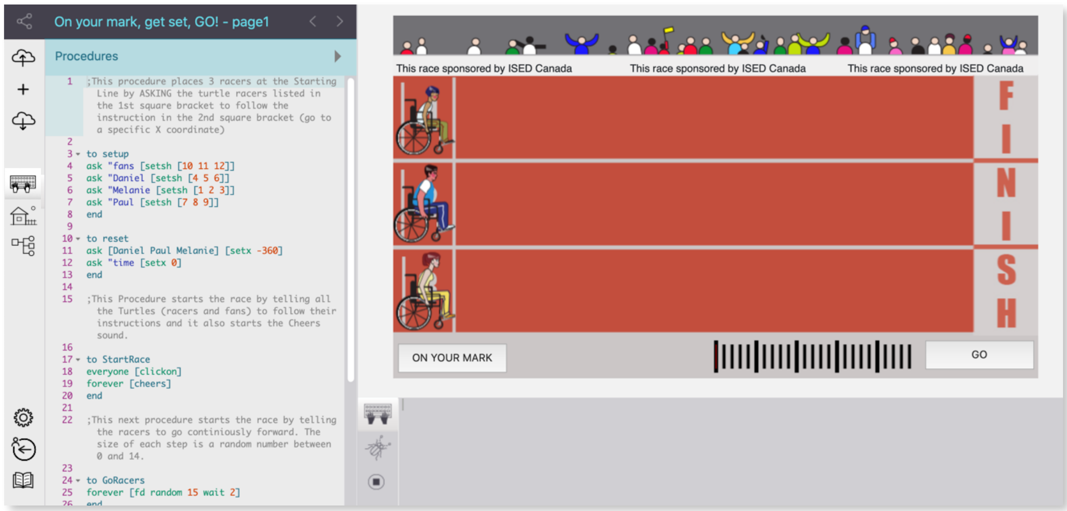
Created at: 2019-06-13 01:21:20

Description: Watch the Race and learn how to code animations and collision detection

Tags or keywords: Animation, Collision Detection

Public Folder: No Folder

The project opens in the Lynx editor.


On your mark, get set, GO! - page1
Procedures
This race sponsored by ISED Canada
This race sponsored by ISED Canada
This race sponsored by ISED Canada
FINISH
ON YOUR MARK
GO

```
;This procedure places 3 racers at the Starting
  Line by ASKING the turtle racers listed in
  the 1st square bracket to follow the
  instruction in the 2nd square bracket (go to
  a specific X coordinate)

to setup
ask "fans [setsh [10 11 12]]
ask "Daniel [setsh [4 5 6]]
ask "Melanie [setsh [1 2 3]]
ask "Paul [setsh [7 8 9]]
end

to reset
ask [Daniel Paul Melanie] [setx -360]
ask "time [setx 0]
end

;This Procedure starts the race by telling all
  the Turtles (racers and fans) to follow their
  instructions and it also starts the Cheers
  sound.

to StartRace
everyone [clickon]
forever [cheers]
end

;This next procedure starts the race by telling
  the racers to go continiously forward. The
  size of each step is a random number between
  0 and 14.

to GoRacers
forever [fd random 15 wait 2]
end
```

While playing with this project, think about what you already know how to do and what's new. Just look at the project and consider what you like about it and what you would probably do differently.

Step 3: Plan your work

Although it's exciting to jump in and begin a new project, it's important to start by first focusing on your game design. There are several reasons why you should think about your design before you start coding.

The end project looks (and works!) better when you start with a plan and you know what your goal is.

You don't want to spend a long time creating some code only to change your mind. A good way to build a project is to start by creating a first draft of the project that includes the main project components. Then go over the project several times, each time adjusting and fine-tuning it to best achieve your original goals. This is what is referred to as an iterative approach to development.

You can manage your time better. When planning, remember to include some time at the end for polishing the final project.

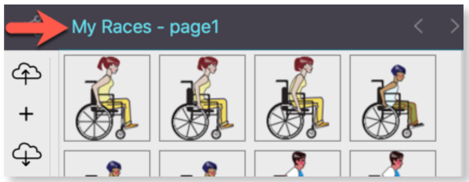
Lesson 2 - On Your Mark, Get Set, GO!

Step 1: Set Up Your Workspace

Click on All Projects on the Lynx home page (www.lynxcoding.org).

In the "Templates" folder, select the "Races Template" project. This project is like a new project except it contains a special set of sample clipart you may want to use for your racing simulation. Click on Edit and the project will open. Click on the "House" icon on the left side and you will see the sample clipart that is included.

RENAME and **SAVE**: Give your project a new name by clicking on "Races Template - page1" above the Procedures Pane and typing in a new name — e.g., **My Races**



Click Save (Arrow to Cloud icon). Lynx creates your own copy of the "Races Template" project in your personal area with its new name — e.g., **My Races**

It is important to remember that Lynx ***does not automatically save your work***, so you must remember to ***save your projects often***!

Step 2: Create a Background

There are two options to create a background in Lynx:

Method 1: Paint the work area with a turtle.

You can use the turtle to draw a simple background, with one colour for the race field and a second colour for the finish line area.

Start by changing the turtle's colour. Type this in the Command Centre:

`setcolour "green`   `Setc` is the short form of `setcolour`.

Try different inputs to find the colour that is right for you!

Other colour names that you can use in Lynx are black, gray, white, red, orange, brown, yellow, green, lime, sky, cyan, blue, magenta, turquoise, violet and pink. Besides colour names, Lynx accepts numbers from 0 to 139: setcolour 55 will do the same as `setcolour "green`. Colour family numbers come in tens. In the Help Section, in the User Guides, there is a Colour Chart.

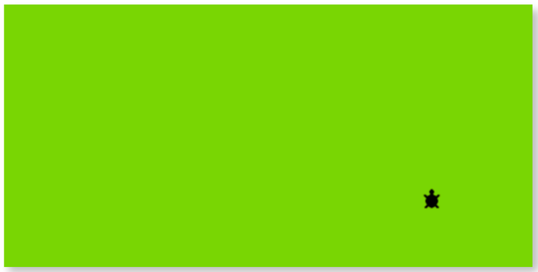
Next, in the Command Centre, type:

```
fill
```

The turtle fills the page, or any enclosed area in which it is placed, with its colour. Since the turtle is the same colour as the background, it doesn't show. Set the turtle to a different colour to see where it is. Run this:

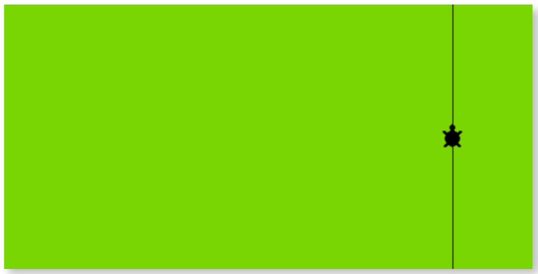
```
setc "black
```

To create an area with a different colour, first, drag the turtle to where you want the finish line to be.



Next, put the turtle's pen down:

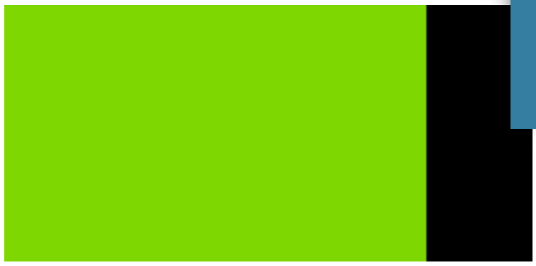
```
pd
fd 1000
```



The number that you enter should be big enough to ensure the line the turtle draws will wrap around the screen, leaving no gaps.

Then move the turtle to the area that you want to colour in some colour different from the main field colour and `fill`.

Can't see the turtle again? Remember to set the turtle to a different colour to see where it is.



Method 2: Use clipart to create a background.

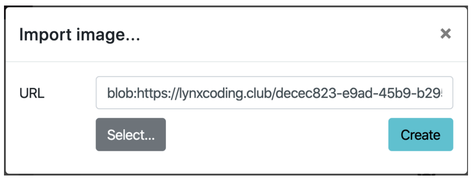
First, find some clipart:

- Use any paint program and draw your own image. Save your art as a JPG or PNG (PNG supports transparency around your objects if you need it).
- Download images from any image search web site or a photo you've taken. Respect copyright issues!
- Use the camera on your smartphone or tablet.

Next, to add your image to the Clipart Pane, either:

- Copy the clipart. Then click on the House icon to open the Clipart Pane. Click on an empty box to reveal a "+" sign. Next press Ctrl-V (Command-V on a Mac) to paste your image in the box.
  Or:

- Click on an empty box in the Clipart Pane and click on the "+" that appears. Then use the dialog box to locate a clipart file on your device or online. Finish by clicking the Create button.



Once you have added your clipart to the Clipart Pane, follow these steps to create a background:

1. Put the mouse over the background image in the Clipart Pane to see the shape number.

   In the Command Centre type:
   `setshape 28` Use whatever number your shape is.

   Press Enter.

2. Move, the turtle to centre the image. In the Command Centre, type:
   `stamp`

   In case you change your mind about the background you created, type:
   `cg` or `cleargraphics`

Set the turtle back to its original shape:

```
setsh 0
```

This will set the turtle to its original shape. Although the turtle's pen may be green (or any other colour) no matter what shape it's set to, you can only see its colour when it is in its original shape.

Step 3: Create and program a turtle

Next, create the racers. Here is a hint: program one racer first and, once you have added all the features and tested the racer's behaviour, duplicate it to create your starting line up.

Select Turtle in the "+" menu.

Turn the turtle to face the right side of the screen (towards the finish line). You can do this in any of these 2 ways.

1. In the Command Centre, type:

   ```
   rt 90
   ```

   If the turtle is pointing straight up, this works, but if the turtle is headed in any other direction, it won't be pointing exactly to the right. The turtle turns right 90 degrees relative to where it was initially pointing.

2. Type:

   ```
   seth 90
   ```

   `Seth 90` always turns the turtle to face east, no matter which way it was pointing before.

Next, animate the turtle.

For your project you can either use the sets of shapes already included in the Races Template or download images from any image search web site. Respect copyright issues!

For example, if using the 3 shapes for the girl in the wheelchair in the sample clipart, first click on the House icon and open the Clipart Pane.

Find the three shapes. Put the mouse over the first shape image in the Clipart Pane to see the shape number.

Then check the shape numbers for the rest of the images of the girl.

Lesson 2 - On Your Mark, Get Set, GO!

In the Command Centre type:

`setshape [1 2 3]` Use whatever numbers your shapes are.

This creates a process that rotates through the shapes in the square brackets as the turtle moves.

When using multiple shapes, always include the list of shape numbers in square brackets with a space between each number.

To test it, type:

`penup` This avoids a line being drawn in the work area. `Pu` for short.

`forever [forward 5]`

This creates a second process. Both processes, changing shapes and moving forward, run at the same time. Moving too fast? Add a `wait 1` after `forward 5`.

Forever means repeat forever whatever commands are in the square brackets. It takes only one input – the list of commands that must always be in square brackets. To stop a forever command, type: stopall

Or click on the square stopall button to the left of the Command Centre.

Now, give your racer a meaningful name. It is always a good idea to name your turtles so that you know who does what (and which turtle-racer wins).

Right-click on your racer. Type whatever name you think is appropriate in the Name field, for example, “Elodie.”

Now, get “Elodie” to continuously move forward. Click on the Keyboard on the left side of the Lynx screen to open the Procedures Pane and define a racing procedure:

```
to GoRacers
forever [fd 7 wait 2]
end
```

Right-click on Elodie and in the On click field drop-down menu, choose `GoRacers`. Click Apply. Now move Elodie to the left side of the screen and click on her to see her move towards the Finish line.

Note: we are using the forever primitive because you want your racer to go on and on -- until it reaches the finish line. The `wait` command slows down the movement a little bit.

Step 4: Save your Project!

Lynx does not automatically save, ***so you must remember to save your projects regularly***.

Lesson 3 – Winning the Race

Step 1: Creating a finish line

Elodie is the model you'll use when creating your other racers, but before you create them, you have some more programming to do.

How will the racers know when the race is over? In this project, you'll use an On Touch event handler which occurs when one turtle collides with another.

To begin, you need another turtle that will act as your finish line. Select Turtle in the "+" menu and give it a name such as `finish_line`.

In Lynx, object names are also used as commands. Turtle names must be one word, with no spaces.

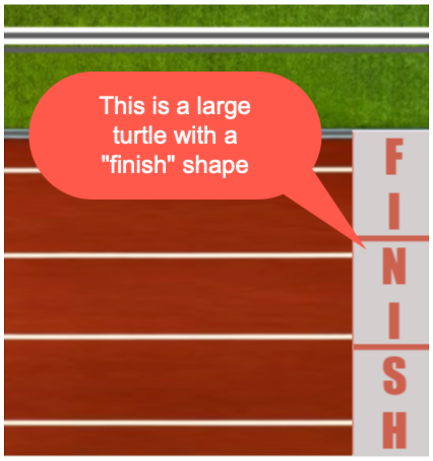
Set the turtle to an appropriate finish line shape, using either the clipart already in the Races Template or clipart you import.

The finish_line turtle should be large enough to reach from the top to the bottom of the screen (or large enough to make sure multiple racers can touch the finish line).

To enlarge the turtle, type:

```
finish_line, setsize 60
```

Try different size inputs. You can choose a different name for the finish line turtle.

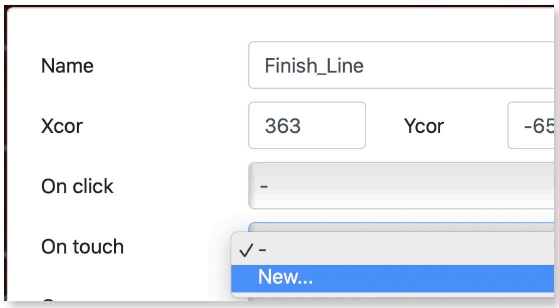


Step 2: Colliding

A collision event handler triggers a procedure to run only when a turtle touches another turtle. The handler requires one input, a variable that reports the name of the turtle that, when touched, triggers the event.

Lesson 3 – Winning the Race

Right-click on the `finish_line` turtle that is in the workspace. Click on the down arrow in the box next to "On touch" and select "New" in the drop down menu.



Click Apply and then Click on the Keyboard icon on the left side to open the Procedures Pane, you'll see:

```
to finish_line_touch :touchedturtle
; Use instructions like these to set this
  turtle's reaction when it touches
  another turtle.
; The variable :TOUCHEDTURTLE contains
  the name of the other turtle.
; SAY "OUCH!
end
```

Remember, any line beginning with a semi-colon (;) is a comment and is not code. Add your own comments before, after, or in the middle of your program to explain what it does or to add additional information.

Change the code in the Procedures Pane to this:

```
to finish_line_touch :touchedturtle
say word 'the winner is' :touchedturtle
announce word  'Congrats' :touchedturtle
stopall
end
```

A colon (:) indicates a word is a variable that reports the name of whatever turtle touches the finish_line turtle each time the program runs.

The `finish_line_touch` procedure uses `announce` and `say`. `Announce` displays the word created by the word primitive in an alert box. The `say` command makes the computer say the word created by the word primitive.

These commands each require a single input that, in this project, is a text string or word, (which is what a text string is called in Lynx). The primitive word is used to create a single word from multiple inputs. It then reports this single word to whatever command it's reporting, in this case, either `say` or `announce`.

Lesson 3 – Winning the Race

There are a few ways to indicate that something is a word:

- Use a single quotation mark (`'`) at the beginning and end of a text string.
  `'Hello, world'` is a text string (or, in Lynx, a word) with 12 elements (the seventh element is a space). In the Command Centre, type:
  ```
  announce 'Hello, world'
  ```
  An alert box opens with:
  Hello, world
- Use a double quotation mark (`"`) at the beginning of a text string. This is used when a text string doesn't contain special symbols (space, etc.). For example, `"Elena` is a text string (word) containing 5 letters or characters.
  ```
  show "Elena
  Elena
  ```
- If it's output by the primitive word, which itself requires two or more inputs that are text strings. Try:
  ```
  say word 'hello, hello' "Elena
  ```

If you don't hear anything, make sure your sound is on.

So, in the procedure line …

```
say word 'the winner is ' :touchedturtle
```

… `word` combines `'the winner is '` and `:touchedturtle` into one word and reports this word to the command `say`.

The instruction line for `announce` works in the same way:

```
announce word  'Congrats ' :touchedturtle
```

`Word` combines `'Congrats '` and `:touchedturtle`.

Use your imagination to choose a reaction on the winner's finish.

Step 3: Here Come the Clones

Once you're sure that the Elodie racer moves as you want and reacts to the finish line, you can clone her and make as many racers as you want do the same thing as Elodie.

In the Command Centre, type:

```
clone "Elodie
```

Please note that we use quotation marks before the word Elodie. This tells Lynx that Elodie is an input for `clone`.

Repeat the `clone` instruction one or two more times to add more racers. Give each racer a unique, meaningful name such as those in the example below Max and Matto.

Step 4: Set-up your game

If your other racers are going to use some different sets of shapes, it's a good idea to write a setup procedure.

Use your racers' names and shape numbers

```
to setup
ask "Elodie [setsh [1 2 3]]
ask "Max [setsh [4 5 6]]
ask "Matto [setsh [7 8 9]]
; sets all the racers to their wheelchair racer shapes
end
```

Type `Setup` in the Command Centre.

Step 5: Save your Project!

Lynx does not automatically save, ***so you must remember to save your projects regularly***.

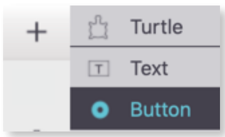
Lesson 4 – All together now!

Step 1: All together now!

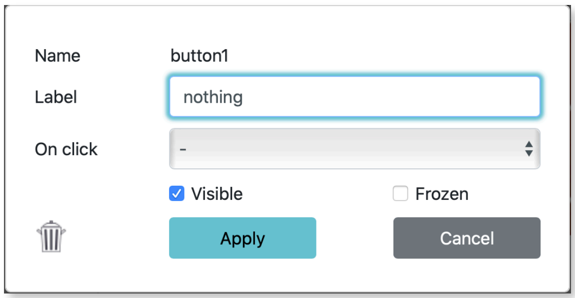
Line up all your racers at the starting line. You could click on each one to start the race, but that would mean they aren't starting at *exactly* the same time as there is a small delay between your clicks. Instead, in the Command Centre, type:

```
everyone [clickon]
```

You can make a button for launching the race.



1. Select Button in the "+" menu.
2. Right-click on the button. You'll see:


Name button1
Label nothing
On click -
Visible ✓
Frozen

3. Add a label. It can be anything, and one or several words. This is just a button label, not a command. Type something that lets you and your project viewers know what the button does, for example, START or GO.
4. Click on the down arrow in the box next to On Click and select New because you don't yet have a procedure for the button to run.
5. Click Apply.

You should see a new procedure in the Procedures Pane:

```
to button1_click
; Use an instruction like this when the button is clicked:
; FORWARD 100
; If the action takes a long time, click on the button again
  to stop this action.
end
```

Edit the procedure to read:

```
to button1_click
setup
everyone [clickon]
end
```

Now, click your button and see the race begin!

Lesson 4 – All together now!

Step 2. Make the race unpredictable

In the `GoRacers` procedure, the input for `fd` was a specific number (`fd 7`), so everyone moved at the same speed. To make a real, unpredictable race, add some randomness.

Change the input for the forward command in GoRacers procedure:

```
to GoRacers
forever [fd random 7 wait 2]
end
```

`Random` takes a number as an input and reports back a randomly chosen (non-negative) integer, including zero, that is less than its input. For example, `random 7` would randomly pick 0, 1, 2 , 3, 4, 5 or 6 and report it to another command.

Certain primitives trigger actions, others report information (a word or number, for example) to another primitive. Primitives that report include word and random. They must report to another primitive and can't stand alone.

For example, if you type

```
random 10
```

….you'll get the message

```
I don't know what to do with 5
```

`Fd 7` moves the turtle forward 15 pixels, but with `fd random 7`, the size of each step is a random number between 0 and 14.

START Click on the GO or START button to start the race.

The racers should move at different rates because random picks a new number every time the instruction is run. Try different inputs for `random` (and `wait`) until you find a nice range of motion.

Step 3: Set the starting position.

It’s good practice to make your race reusable so you can relaunch the race over and over.

To do this, you need instructions to move each racer back to its starting positions. The simplest way to control a turtle’s position is to use its coordinates.

Every position in the Work Area has both an x-coordinate and a y-coordinate. The centre point of the page is position [0 0]. You can find out a turtle’s coordinates with `pos`, `xcor` and `ycor` commands and change its coordinates with `setpos`, `setx` and `sety` commands.

Drag “Elodie” to the left edge of the screen to a place you would consider a good starting line position. Be sure “Elodie” is entirely on the page.

In the Command Centre, type:

```
Elodie, show pos
```

You’ll get a response like this:

```
-360 -9
```

The first number is the x-coordinate, the second the y-coordinate.
Your numbers may be different.

Move all the other racers to their starting line positions. To make sure that the x-coordinate for each turtle is the same, in the Command Centre, type:

```
ask [Elodie Max Matto] [setx -360]
```

You may want to space the racers evenly along the y-axis by setting each turtle's ycor (y-coordinate) appropriately. You may also want to check that each turtle's heading is set at 90 so that it will run in a straight line towards the finish line.

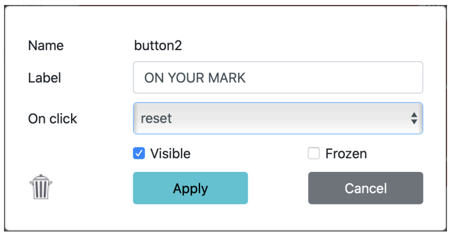
You will use these positions to put the racers at the starting line before each race.

```
to reset
ask [Elodie Max Matto] [setx -360 seth 90]
end
```

The `ask` command asks the turtle (or the turtles) to run the instructions included in the square brackets. If you ask a list of turtles (a list must be included in square brackets) to run the instructions, the first turtle of the list runs the instruction, then the next turtle does the same only after the first turtle is done, and so on.

Lesson 4 – All together now!

When running this procedure, your racers should move to their start positions.



Name button2

Label ON YOUR MARK

On click reset

☑ Visible ☐ Frozen

Make an ON YOUR MARK button that sets every racer to its start position.

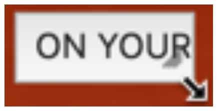
Go to the left side of your screen, click on the "+" sign.

Click on Button and when the Button is on the screen, right-click on it.

Change the Label to ON YOUR MARK , set the On click instruction to Reset and Click Apply.

Move the button to an appropriate place on your screen by dragging it.

Tip: You can change the size of the button by hovering your cursor over the button and adjusting in the bottom right corner.



Step 4: Save your Project!

Lynx does not automatically save, so you must remember to save your projects regularly.

Lesson 5 – Polish Your Game

Step 1: No cheating! Freeze the racers

Most probably, you don't want your finish_line to be draggable at all. You can freeze it to the correct location, so it can't be dragged.

In the Command Centre, type:

```
freeze "finish_line
```

This way you'll make sure it won't be moved.

You may also want to freeze all the racers in your game and the buttons in your project:

```
freeze "button1
```

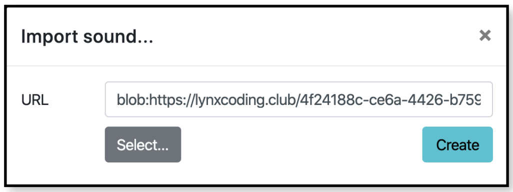
Freeze does not mean that the turtle can't move. After freezing, the turtle obeys commands perfectly, it just can't be dragged by the mouse.

When you need to drag your button, unfreeze it:

```
unfreeze "button1
```

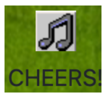
Step 2: Adding Sound and Music

To bring a WAV or MP3 sound file into your project, choose Sound in the "+" menu. The Import Sound dialog box appears. Choose a sound file on your device or online and click Create.



Import sound...

URL: blob:https://lynxcoding.club/4f24188c-ce6a-4426-b759



A sound icon appears on your page.

Add your music or sound to one of your procedures. If you want the sound to repeat continuously, use the `forever` primitive. For example, add a cheering sound to your `StartRace` procedure.

```
to StartRace
everyone [clickon]
forever [cheers]
end
```

Assuming you have a sound named CHEERS.

Test it. Does the sound occur at the right time? Edit your procedure until you're satisfied.

Add additional sounds where appropriate in your story -- at the end of your race. You may even want to use different sounds at the finish line depending on who finished first.

Lesson 5 – Polish Your Game

Step 3: Adding Features

What other features can you add to your project? What about a cheering fan or a whole crowd of them? Or signs in the race area? Or a first page that describes the race?

You may want to add a billboard that flashes during the race by adding a text box and a program that runs when the race starts.

Add a timer. Look at the `timer` procedure in the sample. Can you figure out how it works?

When adding turtles, text boxes, or any actions, make sure you include any instructions for them to act (for example, jump up and down), in the appropriate procedure. Think carefully about when you want each action to start and when you want it to stop.

Remember, the `stopall` command in the `finish_line_touch` procedure will stop all actions and sounds.

Use Your Previous Knowledge! You may want to bring some of your skills from the Story Coders' project into this project. Think about creating a story to wrap around the race by making multiple pages before and after the race page.

Step 4: Save your Project!

Lynx does not automatically save, ***so you must remember to save your projects regularly*****.**

Save often! It's no fun to work on a project and then lose it by forgetting to save.

Lesson 6 – Complete your Game and Play with it

Step 1: Making Observations

Here are a few questions to consider:

1. Is random really random?

   Play with the game you created. Does each racer have an equal chance to win?

   Keep track of how many times each racer wins. Are they all equally likely to win? How many races do you need to run to accurately determine the distribution of wins? Is 3 enough? Is 15 enough?

2. Next, consider these questions:

   - If two racers have the same speed, will they always finish simultaneously?
   - If one racer is moving forward random 10 and another moving forward random 5, will the first racer always win? What if you adjust the input to wait?
   - If one racer moves forward random 10 and another moves forward random 5 + random 5, does one racer have an advantage?

To test these ideas, you'll need to create different racing procedures for each racer, instead of every racer running the `GoRacers` procedure.

This is what you had:

```
to GoRacers
forever [fd random 15 wait 2]
end
```

Your new procedures may look like these:

```
to GoElodie
forever [fd random 10 wait 2]
end

to GoMax
forever [fd random 15 - random 5 wait 2]
end

to GoMatto
forever [fd random 5 + random 5 wait 2]
end
```

Assign each one to the appropriate racer's "On click" instruction.

Keep track of how many times each racer wins. Is there a difference in win rates?

Lesson 6 – Complete your Game and Play with it

How else can you adjust the procedures to test different ideas?

You may want to add a page to your project noting your observations.

Step 2: Save your Project!

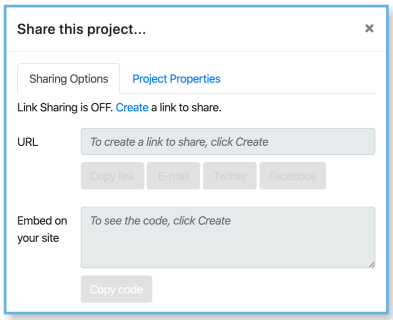
Lynx does not automatically save, ***so you must remember to save your projects regularly***.

Step 3: Share your Project.

When you've finalized your game and are ready to go public, there are several ways to share your project. You should talk to your teacher about the best way to share your projects.

From the Lynx editor

- From within the Lynx editor, simply click on the Share icon in the top-left corner of the editor.
- On the Sharing Options tab, click Create a Link. Then click on one of the Share sites (Twitter, Facebook) or copy the link to paste it where needed or click on E-Mail to send the link by email.
- You can choose an image file to use as a preview. Click the Project Properties tab on the Dialog Box (use the buttons to get a file from your device).



Share this project... ×

Sharing Options | Project Properties

Link Sharing is OFF. Create a link to share.

URL: *To create a link to share, click Create*

Copy link | E-mail | Twitter | Facebook

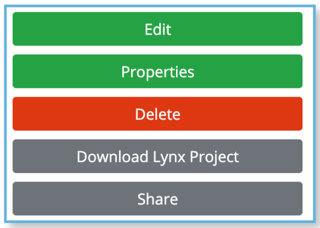
Embed on your site: *To see the code, click Create*

Copy code

From within your Lynx personal space in the Cloud

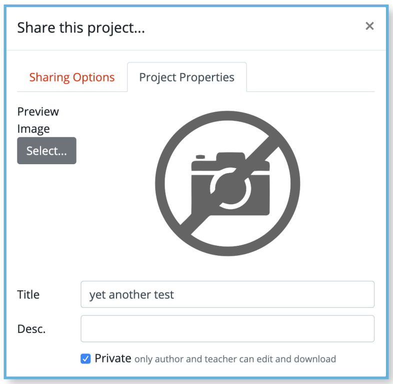
- From your Lynx personal space, click on your project to open it in Play Mode.

- Then click on Share. Follow the instructions as above.



Enjoy (and edit) a copy of my project

- You can also let your friends copy and edit the project.
- Before sharing a project, go to its Project Properties tab.
- Uncheck the Private check box.
- Click on the Sharing Options tab and copy the link.
- Send the link to a friend. Using the link, he / she will be able to make and save changes to a copy of your project.



Share this project... ×

Sharing Options | Project Properties

Preview Image: Select...

Title: yet another test

Desc.:

☑ Private only author and teacher can edit and download