

[CODECYPRUS 2014]

Φτιάξτε το δικό σας Frogger παιχνίδι  
Build your own Frogger mobile game

Nearchos Paspallis  
Νέαρχος Πασπαλλής

# What will we be doing today

## Learn what programming is

- What can a computer **really** do?
- How to have the same operations **repeat again and again**
- How to tell the computer **how to make a decision**
- Combine these to **build a Frogger mobile game**

# How smart are computers?

Not as much as you might think!

Core components of any computer:

- **Central Processing Unit (CPU)**
- **Memory**

Typical instructions are very simple (and mechanical)

**Copy** the contents of one memory location to another

**Add** the contents of a memory location with another's

**Multiply** the contents of a memory location with another's

Etc.

**Computer = Powerful + Stupid**

## But computers do amazing things...

Well they do.

But only because some smart people tell them **how** to do it!

The programmers write **programs** to make this possible

Today, these programmers/engineers are even **cool**!



NASA Engineer (circa 1969)



NASA Engineer (circa 2012)

# What is a program?

“A sequence of instructions that a computer can interpret and execute”

Programs are defined in various programming languages

Today's focus: Scratch

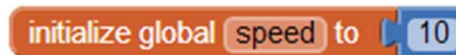
[scratch.mit.edu](https://scratch.mit.edu)



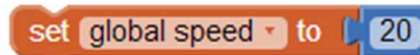
# Let's get started!

One thing computers can do really well, is remembering things

Another thing is doing basic calculations, like addition, subtraction, multiplication, etc.



initialize global speed to 10



set global speed to 20



set global speed to 10 + 10



set global speed to get global speed + 10

For example...

The screenshot displays a software development environment with three main panels: Viewer, Components, and Properties.

- Viewer:** Shows a road simulation with two yellow trucks. The road has a dashed center line and solid edge lines. Below the road, there are four vertical lines with numerical labels: 50, 100, 150, and 200. The trucks are positioned at the 100 and 200 marks.
- Components:** Shows a tree view with 'Screen1' containing a 'road' component and a 'yello\_truck' component.
- Properties:** Shows the properties for the 'yello\_truck' component. The 'X' property is highlighted with a red dashed circle and is being changed from 200 to 100. A code block above the 'X' property reads: `set yello_truck . X to 100`.

Let's do that again

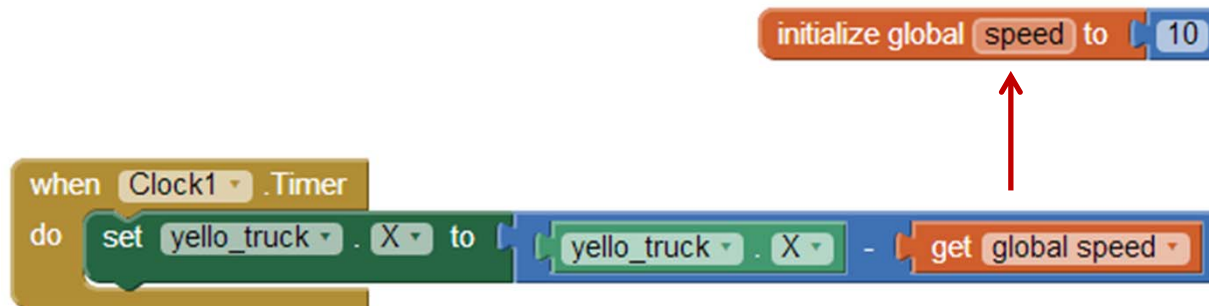
Computers are fast and tireless. They can repeat an operation again, and again, achieving marvellous results.





Let's do that again

Computers are fast and tireless. They can repeat an operation again, and again, achieving marvellous results.



Let's do that again

The screenshot displays a mobile development environment with three main panels: Viewer, Components, and Properties.

- Viewer:** Shows a mobile screen with a road scene. A yellow truck is positioned on the road. The status bar at the top indicates the time is 9:48. A checkbox labeled "Display hidden components in Viewer" is checked. Below the viewer, a section titled "Non-visible components" contains a "Clock1" icon.
- Components:** A tree view showing the hierarchy: Screen1 > road > yello\_truck > Clock1.
- Properties:** Shows the properties for the selected "Clock1" component: "TimerAlwaysFires" (checked), "TimerEnabled" (checked), and "TimerInterval" (set to 100).
- Scripting:** A script block is attached to the "Clock1" component. It contains a "when" block for "Clock1" with a ".Timer" trigger, followed by a "do" block containing a "set" block: "set yello\_truck . X to" followed by a calculation "yello\_truck . X - get global speed".
- Media:** A section at the bottom lists "road.png" and "truck\_yellow\_1.png" with an "Upload File ..." button.

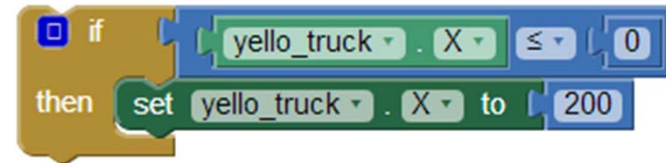
What if...

Another thing that makes computers so powerful, is their ability to analyze a condition and choose a path based on the outcome



What if...

Another thing that makes computers so powerful, is their ability to analyze a condition and choose a path based on the outcome



Let's do that again

The screenshot shows a mobile app development interface with three main panels: Viewer, Components, and Properties.

- Viewer:** Displays a mobile screen with a road scene. A yellow truck is positioned on the road. The status bar at the top shows signal strength, Wi-Fi, and the time 9:48. A checkbox "Display hidden components in Viewer" is checked.
- Components:** A tree view showing the hierarchy: Screen1, road, yello\_truck, and Clock1. Buttons for "Rename" and "Delete" are visible below the tree.
- Properties:** Shows the properties for the selected "Clock1" component: "TimerAlwaysFires" (checked), "TimerEnabled" (checked), and "TimerInterval" (set to 100).
- Non-visible components:** A section at the bottom of the Viewer panel showing "Clock1" as a hidden component.
- Media:** A section at the bottom of the Components panel listing "road.png" and "truck\_yellow\_1.png" with an "Upload File ..." button.

```
if yello_truck . X ≤ 0  
then set yello_truck . X to 200
```

# Time to build our own Frogger mobile game!

We will be using MIT's App Inventor  
[appinventor.mit.edu](http://appinventor.mit.edu)

It is completely Web based

Has two views:

- Designer
- Blocks editor





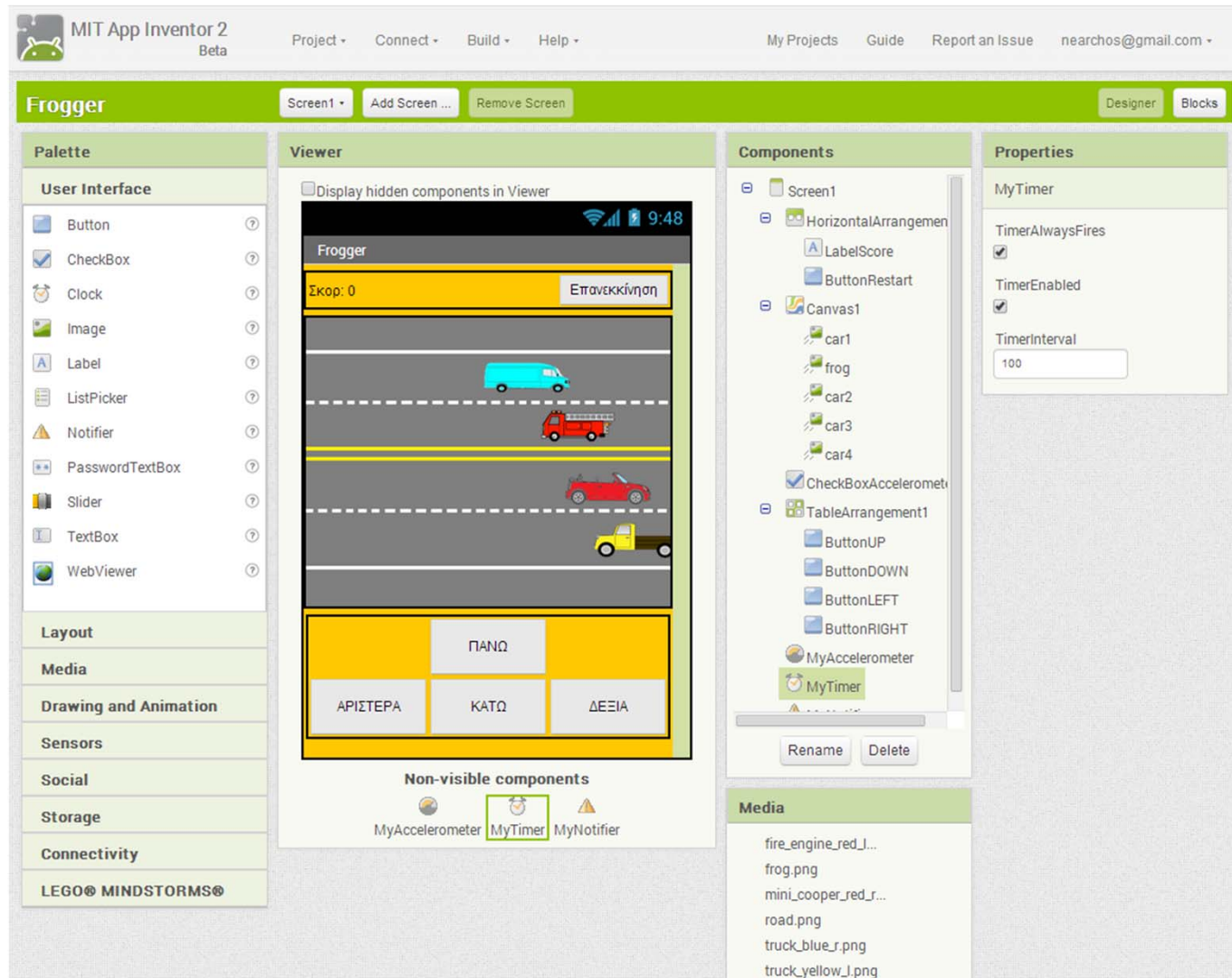
## What is this Frogger game?

- An arcade game by Konami
- What games looked like back in 1981...
- Today we will create our own Frogger game

Continue on the editor ...

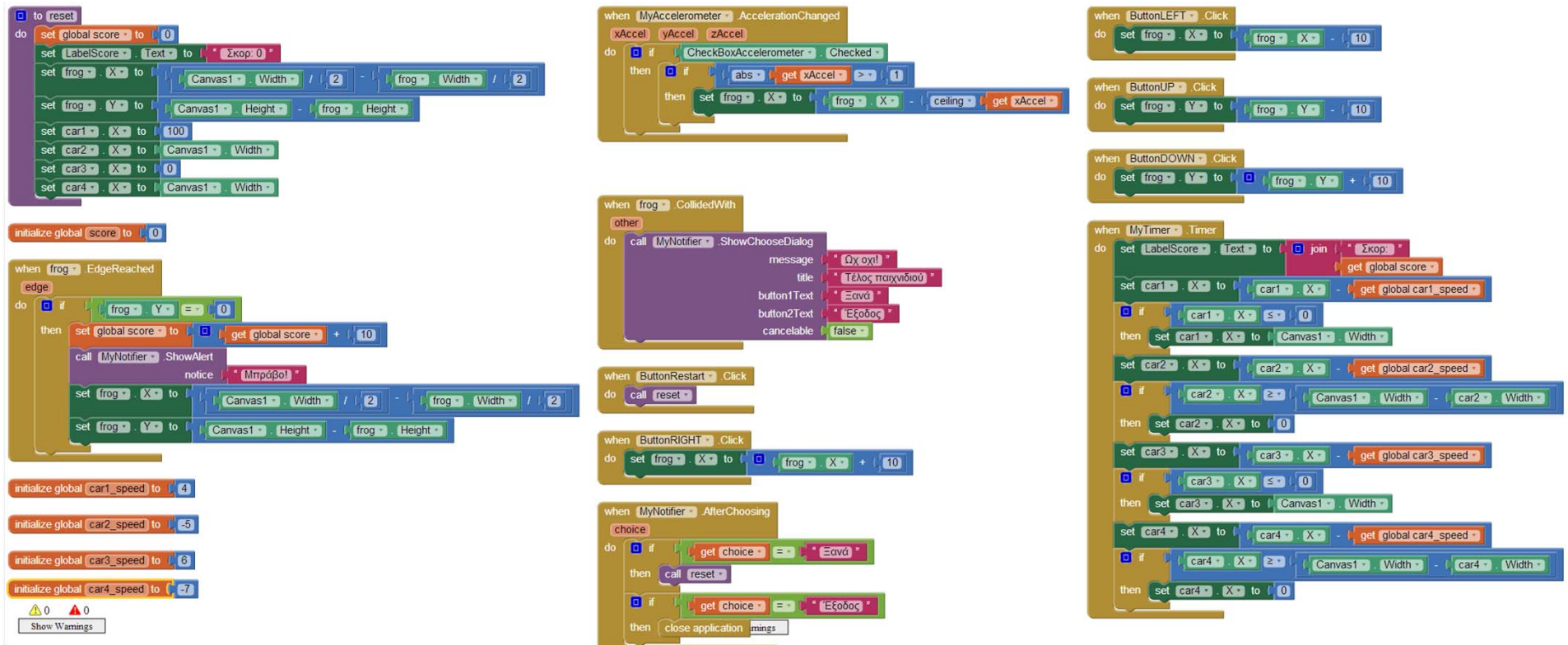
<http://ai2.appinventor.mit.edu>

<http://2014.codecyprus.org/frogger.htm>



Frogger game  
completed  
(design)

# Frogger game completed (logic)



The image displays the logic for a Frogger game, organized into several functional blocks:

- to reset:** Resets the global score to 0, sets the label score text to "Σκορ: 0", and initializes the frog's position to the center of the canvas. It also sets the initial X-positions for four cars (car1 to car4) across the width of the canvas.
- initialize global variables:** Sets global variables for car speeds: car1\_speed to 4, car2\_speed to -5, car3\_speed to 6, and car4\_speed to -7.
- when frog EdgeReached:** Checks if the frog has reached the left edge (Y=0). If so, it increments the global score by 10, shows an alert with the message "Μηδρό!", and resets the frog's position to the center.
- when MyAccelerometer AccelerationChanged:** Checks if the accelerometer is checked. If the absolute value of xAccel is greater than 1, it adjusts the frog's X-position based on the acceleration direction, bounded by the canvas width.
- when ButtonLEFT Click:** Decreases the frog's X-position by 10.
- when ButtonUP Click:** Decreases the frog's Y-position by 10.
- when ButtonDOWN Click:** Increases the frog's Y-position by 10.
- when MyTimer Timer:** A timer that updates the car positions and speeds. It calculates the new X-positions for each car based on their speed and the current time, ensuring they stay within the canvas boundaries.
- when frog CollidedWith other:** Triggers a dialog box with the message "Ωχ οχι!", title "Τέλος παιχνιδιού", and buttons "Ξανά" (Restart) and "Εξόδος" (Exit). The dialog is non-cancelable.
- when ButtonRestart Click:** Calls the reset function.
- when ButtonRIGHT Click:** Increases the frog's X-position by 10.
- when MyNotifier AfterChoosing:** Checks the user's choice. If "Ξανά" is chosen, it calls reset. If "Εξόδος" is chosen, it closes the application.

## Real life lessons

- What happens when a middle aged man rediscovers Frogger?

Thank you!

## Useful resources

- Many, many introductory programming resources @ [code.org](https://code.org)
- Learn by playing with angry birds @ [learn.code.org/hoc/1](https://learn.code.org/hoc/1)
- Build your own Flappy bird game @ [learn.code.org/s/6/level/148](https://learn.code.org/s/6/level/148)
- MIT's App Inventor @ [appinventor.mit.edu](https://appinventor.mit.edu)