# Programming Turtles

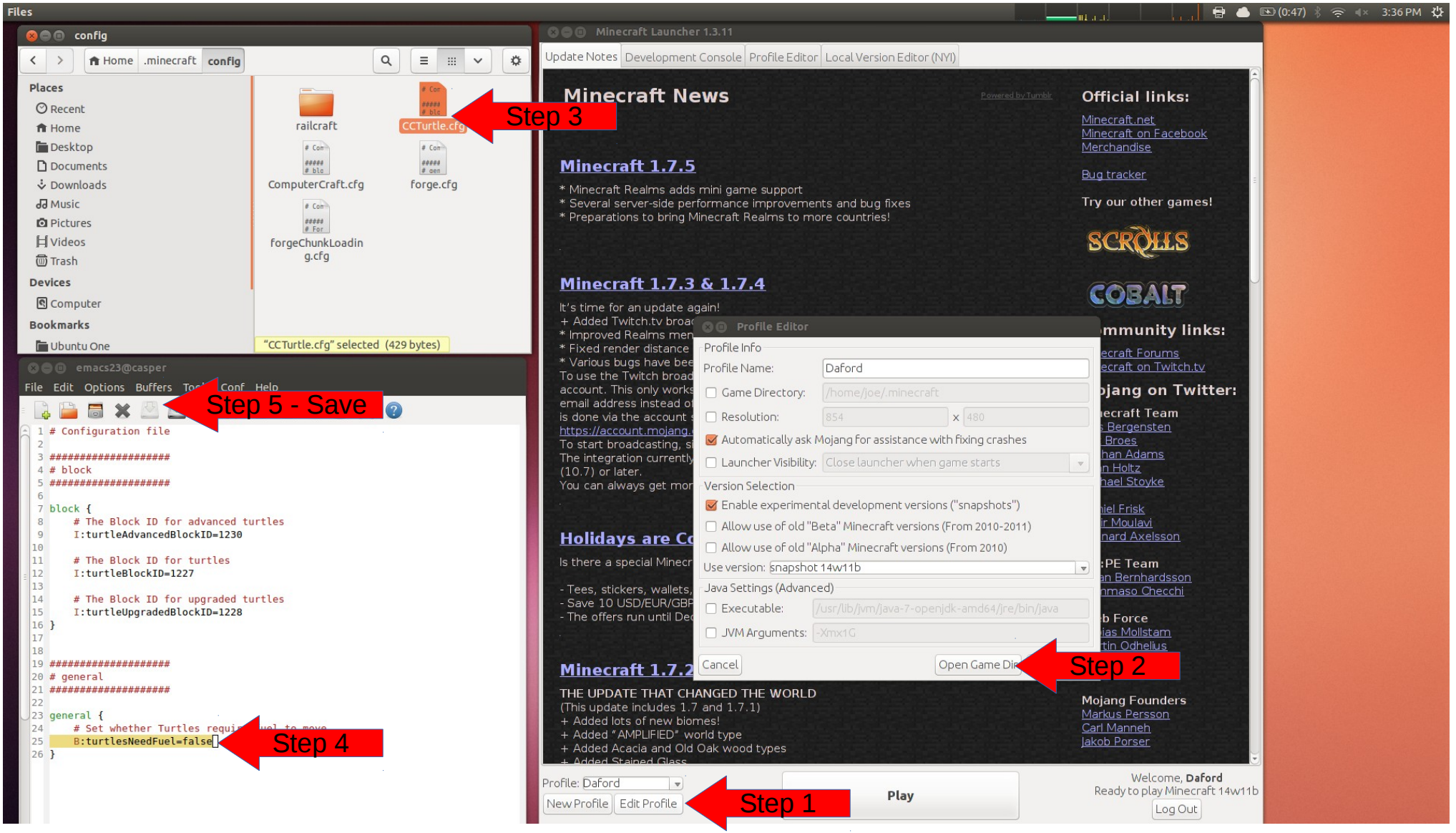
# Challenges

- Program a ComputerCraft Turtle to
  - **Beginner:** craft a basic starter home
  - **Intermediate:** craft a customized house (E.g., two story house, basement, etc)
  - **Advanced:** craft a castle (E.g., circular towers, etc)
  - **General Challenge:** Make a house API (Application Programming Interface)

# Development Environment Setup

- Make turtles work without having to add fuel
- Craft disk drive with disk to save programs
- Learn how to navigate turtles in case something goes wrong!

# Remove Need For Fueling Turtles



# Overview of the Turtle API

- Turtle API (Application Programming Interface) used to make house
  - **turtle.forward()** - Returns true if turtle moves forward, otherwise returns false.
  - **turtle.back()** - Returns true if turtle moves backward, otherwise returns false.
  - **turtle.up()** - Returns true if turtle moves up, otherwise returns false.
  - **turtle.down()** - Returns true if turtle moves down, otherwise returns false.
  - **turtle.turnLeft()** - Returns true if turtle turns left, otherwise returns false.
  - **turtle.turnRight()** - Returns true if turtle turns right, otherwise returns false.
  - **turtle.dig()** - Returns true if turtle breaks block in front, otherwise returns false.
  - **turtle.digDown()** - Returns true if turtle breaks block below, otherwise returns false.
  - **turtle.place()** - Returns true if selected block is placed in front of turtle, otherwise false.
  - **turtle.placeDown()** - Returns true if selected block is placed below turtle, otherwise false.
  - **turtle.detectDown()** - Returns true if a block is below turtle, otherwise returns false.
  - **turtle.select(number slotNumber)** – Make turtle select item from provided slot number. (1 is top left and 16 is bottom right)
- Go to http://computercraft.info/wiki/Turtle_(API ) for complete list of the Turtle APIs

# Introduction to Lua

- **Variables** – Store a changeable value

  local turtleName = “JoeBot”

- **If, Then, Else, End** – Conditional Statement

  if ( turtle.forward() ) then print(turtleName..” moved forward”) end

- **Functions** – Helps simplify code by separating into functionality to prevent repetition

  ```
  local function moveForward(name)
      if ( turtle.forward() ) then print(name..” moved forward”) end
  end
  moveForward(turtleName)
  ```

- **Loops**
  - For Loop

    ```
    for var = start, end, Interval do
        turtle.forward()
    end
    ```

  - Repeat Loop

    ```
    repeat  turtle.forward() until ( turtle.detect() )
    ```

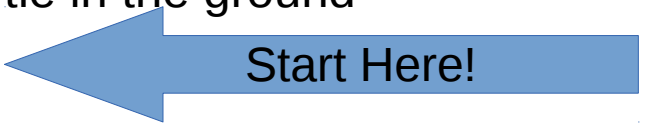
- Go to http://computercraft.info/wiki/Tutorials for a more complete “Basic Tutorial” on Lua

# How to Create a House

- Step 1:  Break down the problem
- Step 2: Start with simplest part of the program
  (E.g., One call to an existing API)
- Step 3: Test to make sure it works!
  (Note: This step validates that your environment works!)
- Step 4: Code next part of the problem
- Step 5: Test to make sure next part works!
- Step 6: Go to Step 4 until house is complete!

# Breaking Down Problem

- Create House
  - Create Floor (E.g., Place many rows of blocks in ground)
    - Place one row of blocks in ground
      - Place one block in front of turtle in the ground
        - Move forward one block
        - Dig one block below
        - Place one block below
        - Return to starting position (E.g., Move back one block)




  - Create Walls
  - Create Ceiling
  - Create Door

# Simple placeBlock() Function

# Better placeBlock() Function

Global Variable

Private Function

Conditional Logic

repeat – until Loop

Exit function

```lua
MAX_SLOT_NUMBER = 16
local function placeBlock()
  if ( turtle.detectDown() ) then
    turtle.digDown()
  end
  slotNum = 0
  repeat
    slotNum = slotNum + 1
    if ( slotNum > MAX_SLOT_NUMBER ) then
      print ("No blocks to place!")
      return
    end
    turtle.select(slotNum)
  until turtle.placeDown()
end
```

# placeBlockRow() Function

Use “for” loop to repeat code

# createFloor() Function

Modulo operator (division remainder)

Helper Function (see next page)

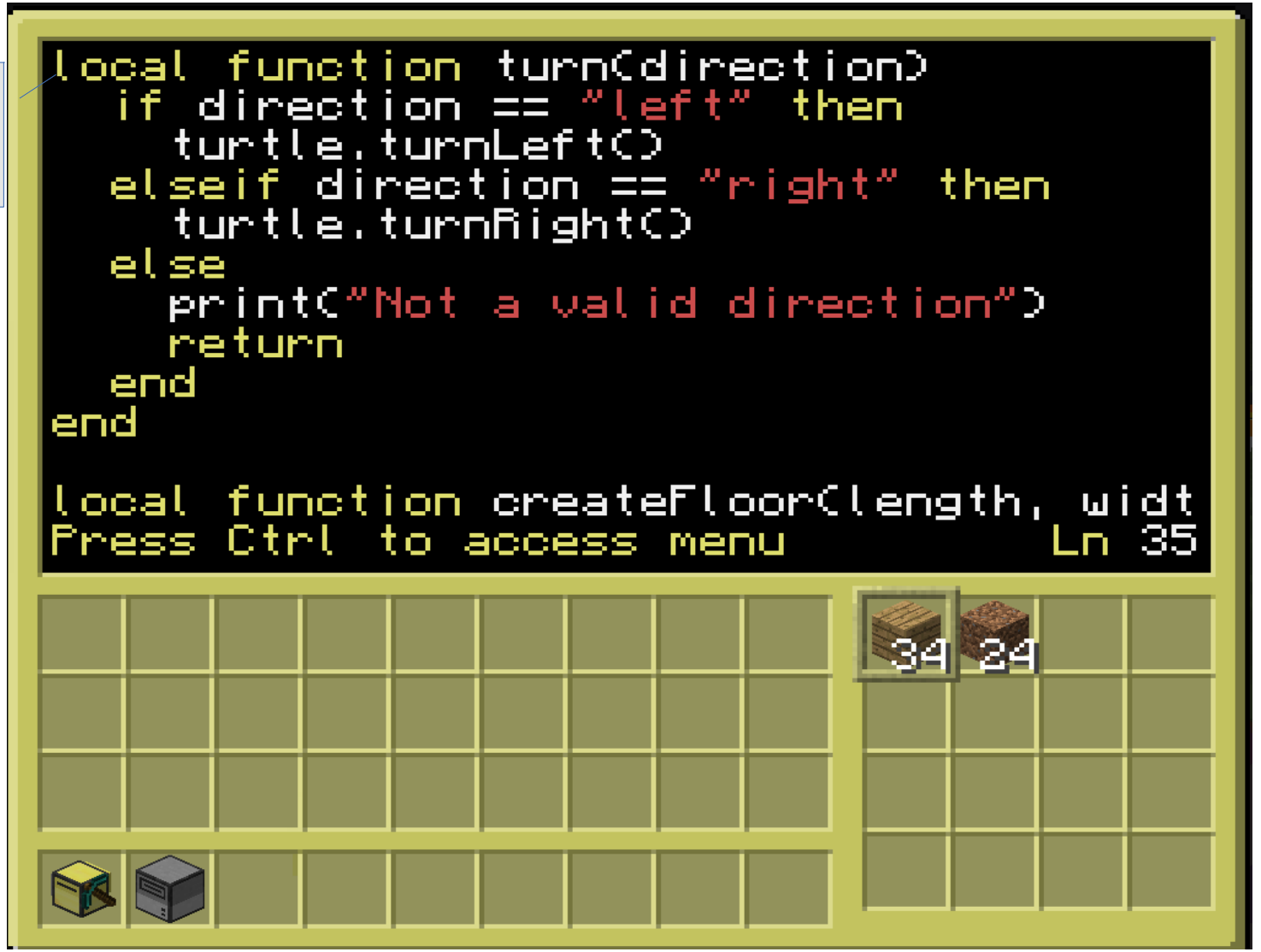
Start to return turtle to starting position

Another Helper Function (see next page)

```
41 local function createFloor(length, width)
42   for i=1, width do
43     placeBlockRow(length)
44     direction = "left"
45     if ( (i % 2) == 0 ) then
46       direction = "right"
47     end
48     turn(direction)
49     turtle.forward()
50     turn(direction)
51     turtle.back()
52   end
53   if ( direction == "left" ) then
54     moveTurtleForward(length+1)
55     turtle.turnLeft()
56     turtle.turnLeft()
57   end
58   turtle.turnRight()
59   turtle.up()
60   moveTurtleForward(width)
61   turtle.down()
62   turtle.turnLeft()
63 end
```

# turn(direction) Helper Function

Helper function must come before function that uses it.



# moveTurtleForward Helper Function

```
35 local function moveTurtleForward(distance)
36   for i=1, distance do
37     turtle.forward()
38   end
39 end
```

# createWalls Function

“for" loop to create four house walls

Helper Function (see next page)

Return turtle to start position

```
78 local function createWalls(length, width, height)
79   for i=1, height do
80     turtle.up()
81     for j = 1, 4 do
82       placeBlockRow(calculateDistance(j, length, width))
83       turtle.turnLeft()
84     end
85     turtle.turnRight()
86     turtle.forward()
87     turtle.turnLeft()
88     turtle.back()
89   end
90   for i=1, height do
91     turtle.down()
92   end
93 end
```

# calculateDistance Helper Function

```
49 local function calculateDistance(index, length, width)
50   if ( index == 1 ) then
51     distance = length
52   elseif ( index == 2 ) then
53     distance = width - 1
54   elseif ( index == 3 ) then
55     distance = length - 1
56   else
57     distance = width - 2
58   end
59   return distance
60 end
61
62 local function createWalls(length, width, height)
63   for i=1, height do
```

# createCeiling() Function

```
 95 local function createCeiling(length, width, height)
 96   for i=1, height+1 do
 97     turtle.up()
 98   end
 99   createFloor(length, width)
100   for i=1, height+1 do
101     turtle.down()
102   end
103 end
```

Reuse createFloor function. The ceiling is equivalent to the top floor

# createDoor() Function

```
105 local function createDoor()
106   turtle.turnRight()
107   turtle.forward()
108   turtle.turnLeft()
109   moveTurtleForward(2)
110   turtle.turnLeft()
111   turtle.dig()
112   turtle.up()
113   turtle.dig()
114   turtle.down()
115   turtle.select(MAX_SLOT_NUMBER)
116   turtle.place()
117   turtle.select(1)
118   turtle.turnLeft()
119   moveTurtleForward(2)
120   turtle.turnRight()
121   turtle.forward()
122   turtle.turnRight()
123 end
```

The door must be in the last slot!

# Putting It All Together With createHouse() Function

```
125 local function createHouse(length, width, height)
126   createFloor(length, width)
127   createWalls(length, width, height)
128   createCeiling(length, width, height)
129   createDoor()
130 end
131
132 createHouse(5,5,5)
```

# Make it an API

- What is an API (Application Programming Interface)
- How to make a house API
  - local (private) vs non-local (public) methods
  - Remove “local” from the following methods
    - createFloor(), createWalls(), createCeiling(), createDoor()
  - Move all code except createHouse() into API file called “house”.  (Refer to API version in Appendix for example)
  - Update makeHouse to use API
    - To use the api call os.loadAPI(“house”).  Now you can just use house.createFloor(), etc.

# More Lua Resources

- http://www.lua.org/manual/5.1/manual.html
- http://coderdojosv.github.io/mobile-games/section-00/docs/introduction.html
- http://repl.it/languages/lua

# Thank You

Twitter: @joeddean

Email: joe.dean@gmail.com

# Appendix