Scratch & Alice Educational Programming Languages

Marc Shelton, Cameron Viezel, Mark Caldropoli, Jasper Suhr

General Overview

- History before Scratch/Alice
- Scratch
- Alice
- Educational Aspects

What is an educational programming language?

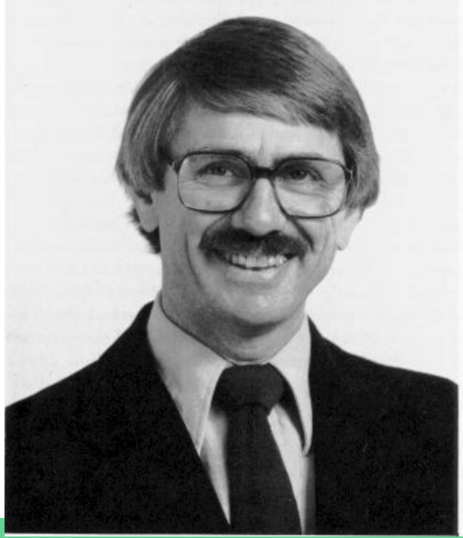
- Designed for learning
- Actual work is an afterthought
- Learning path
- Help children learn how to code
- Not machine code and assembly
- Easy entry for those not exposed to coding before

Before Educational Programming Languages

- High barrier for entry
- Using computers and learning languages was hard
- Only really used by scientists and mathematicians
- Required writing custom software

History of Educational Programming

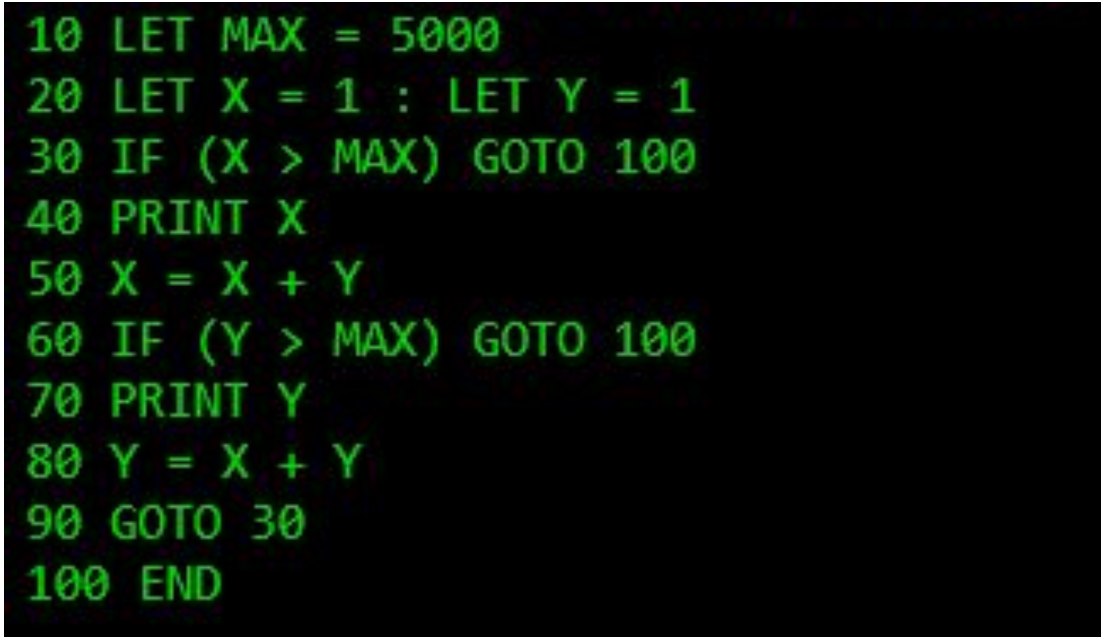
- John G. Kemeny and Thomas E. Kurtz
- Programming literacy outside STEM fields
- DARSIMCO (Dartmouth Simplified Code)
  - Set of macros
- DOPE (Dartmouth Oversimplified Programming Experiment)
- Fortran and ALGOL
- Lack of immediate feedback due to batch processing
- Time-sharing solution



Birth of Basic (1964)

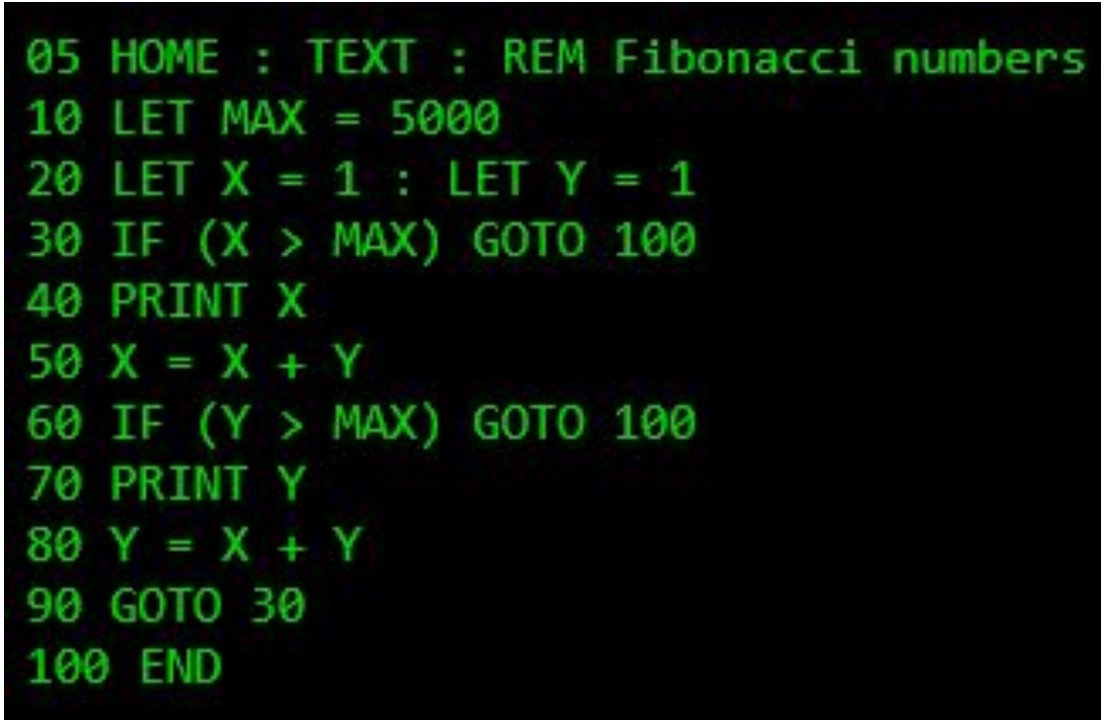
- Beginner's All-purpose Symbolic Instruction Code
- Heavily patterned on FORTRAN II
- Many of the same commands and format but syntax was improved where possible
- DO 100, I = 1, 10, 2 -> FOR I = 1 TO 10 STEP 2
- Mary Kenneth Keller
- Focused on straightforward mathematical work with matrix support, strings added later
- Became extremely popular
- Dijkstra, "It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration"

BASIC Example

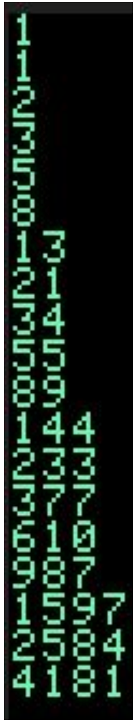


```
10 LET MAX = 5000
20 LET X = 1 : LET Y = 1
30 IF (X > MAX) GOTO 100
40 PRINT X
50 X = X + Y
60 IF (Y > MAX) GOTO 100
70 PRINT Y
80 Y = X + Y
90 GOTO 30
100 END
```

BASIC Example



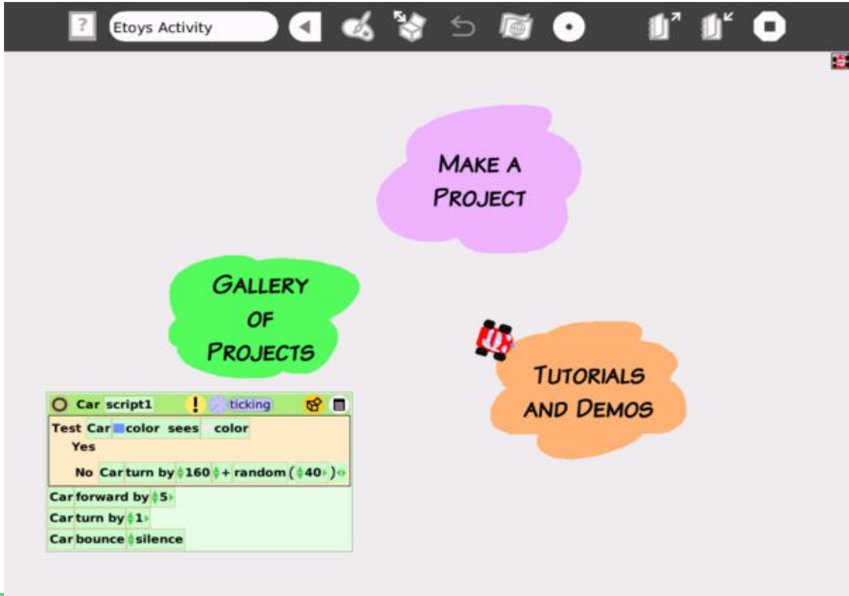
```
05 HOME : TEXT : REM Fibonacci numbers
10 LET MAX = 5000
20 LET X = 1 : LET Y = 1
30 IF (X > MAX) GOTO 100
40 PRINT X
50 X = X + Y
60 IF (Y > MAX) GOTO 100
70 PRINT Y
80 Y = X + Y
90 GOTO 30
100 END
```



```
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
```

Learning Path

- One Laptop per Child project
- Scratch to Etoys to Squeak to any Smalltalk
- Graphical environment to teach kids coding



```
Car script1 ! ticking
Test Car color sees color
  Yes
  No Car turn by 160 + random ( 40 )
Car forward by 5
Car turn by 1
Car bounce silence
```

Squeak

- Implementation of the Smalltalk language
- Application development language
- Squeak was used in Scratch until Scratch 2.0.

Scratch

Scratch Overview

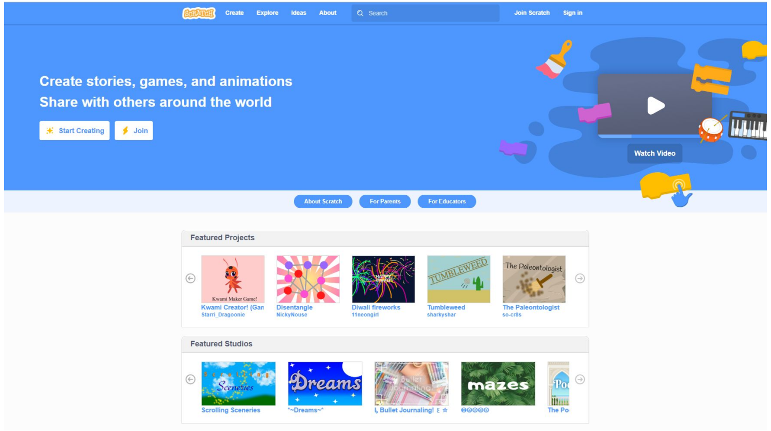
- History of Scratch
- Scratch as a Programming Language
- Error handling
- Block types and their usages
- Variables and lists
- Procedures through custom blocks
- Extensions
- Examples

Scratch History

- Created by Lifelong Kindergarten Group within the MIT Media Lab
- Developed to teach children the fundamentals of programming
- Started development in 2003
- Released version 1.0 in 2007
- Released version 2.0 in 2013
- Released version 3.0 in 2019
- Named ‘Scratch’ after the technique used by DJs to remix songs

Scratch Features

- Popular projects displayed on the homepage
- Online IDE to build and save projects
- Can download project from online IDE as a .sb file
- Offline IDE available for download



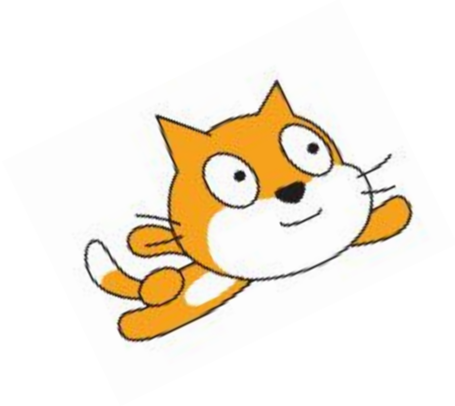
Create stories, games, and animations
Share with others around the world

Scratch as a Programming Language

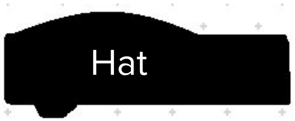
- Dynamically-typed, interpreted language
- Visually programmed by placing sequences of blocks
- Turing-complete (can simulate a Turing machine to run algorithms)
- Primitive data types:
    - Numbers (Floats and Integers)
    - Strings
    - Booleans
- Support for variables and lists
- Code used as a script tied to a sprite
- Lists and variables can be global or local to a specific sprite
- Procedures created with custom blocks

Error Handling in Scratch

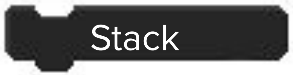
- Large numbers represented in scientific notation
- Number overflow yields ‘Infinity’
- Number underflow yields ‘-Infinity’
- Infinite loops can be terminated with the stop sign
- Ignores errors which would throw runtime exceptions in other languages



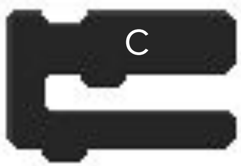
Block Shapes



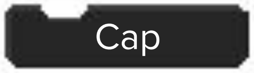
Signifies the start of a script



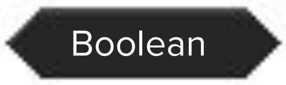
Perform main commands - can be **STACKED**



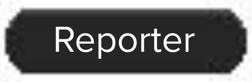
Used for conditional statements and loops



Stop scripts from executing

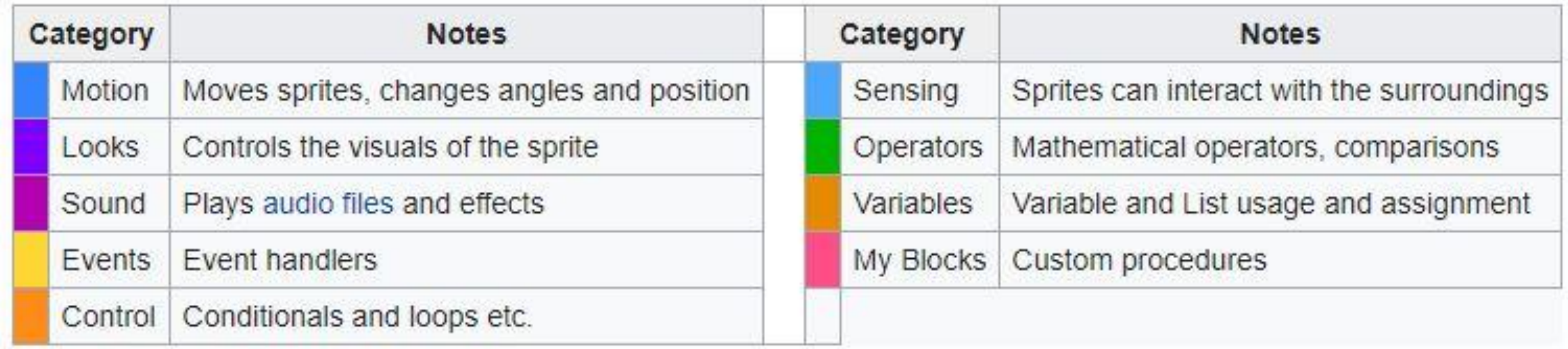


Check True or False



Contain values

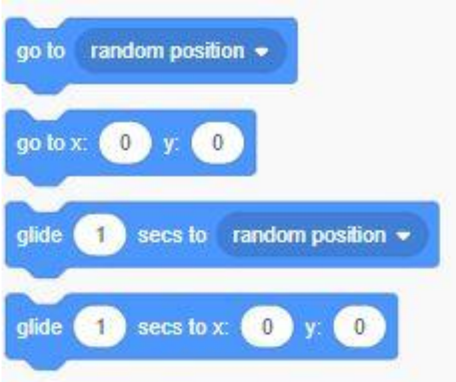
Blocks in Scratch



Category		Notes
🟦	Motion	Moves sprites, changes angles and position
🟪	Looks	Controls the visuals of the sprite
🟪	Sound	Plays audio files and effects
🟨	Events	Event handlers
🟧	Control	Conditionals and loops etc.
🟦	Sensing	Sprites can interact with the surroundings
🟩	Operators	Mathematical operators, comparisons
🟧	Variables	Variable and List usage and assignment
🟥	My Blocks	Custom procedures

Motion, Looks, and Sound

- Move sprites by modifying X, Y coordinate
- Modify sprite and stage appearance
- Play various sound clips

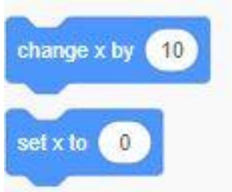


go to random position ▾

go to x: 0 y: 0

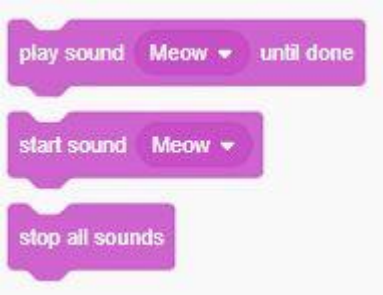
glide 1 secs to random position ▾

glide 1 secs to x: 0 y: 0



change x by 10

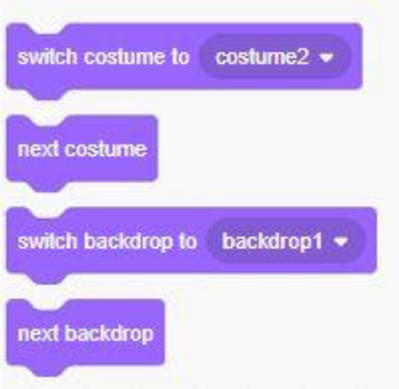
set x to 0



play sound Meow ▾ until done

start sound Meow ▾

stop all sounds

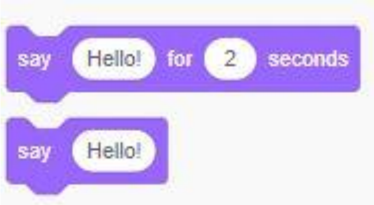


switch costume to costume2 ▾

next costume

switch backdrop to backdrop1 ▾

next backdrop

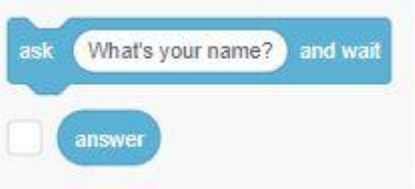


say Hello! for 2 seconds

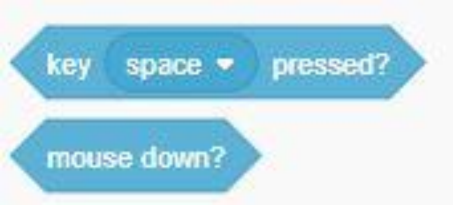
say Hello!

Sensing

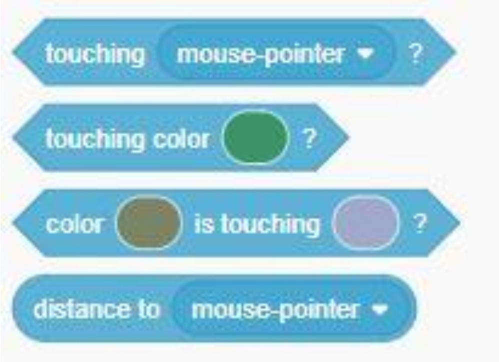
- Detect specific occurrences during runtime



```
ask What's your name? and wait
□ answer
```



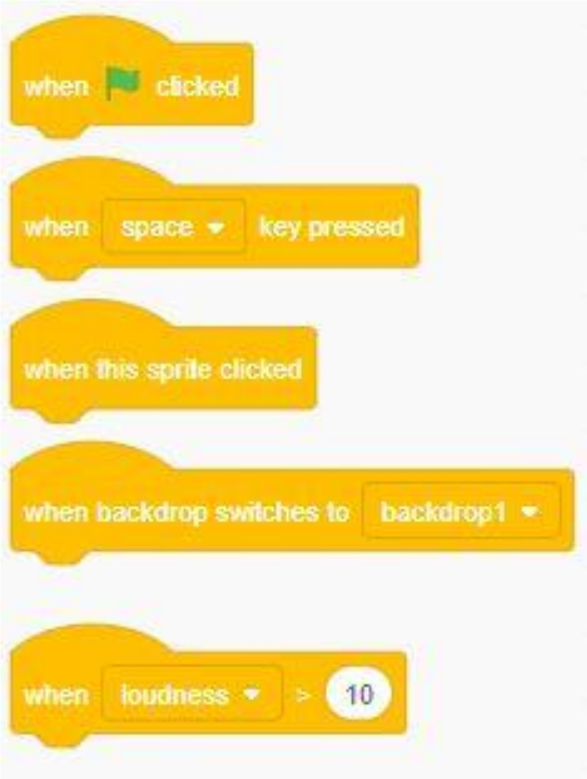
```
key space ▾ pressed?
mouse down?
```



```
touching mouse-pointer ▾ ?
touching color ● ?
color ● is touching ● ?
distance to mouse-pointer ▾
```

Events

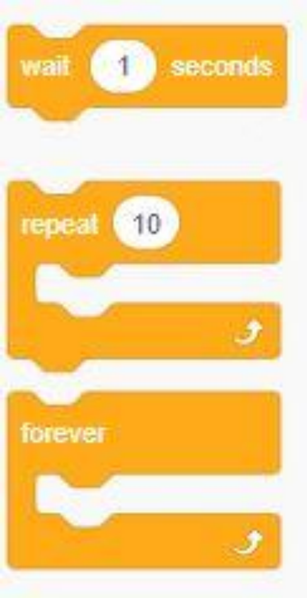
- Trigger scripts to begin execution
- Hat blocks

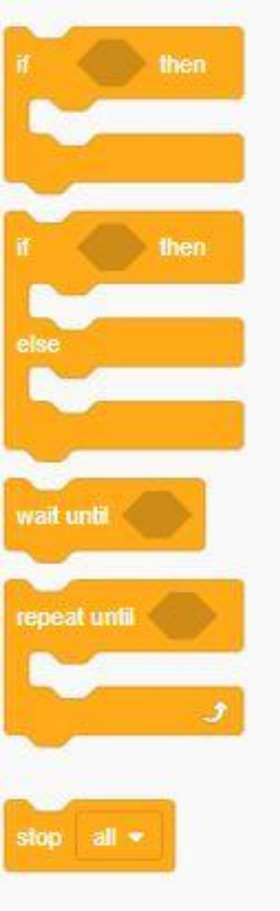


```
when 🏴 clicked

when space ▾ key pressed

when this sprite clicked

when backdrop switches to backdrop1 ▾

when loudness ▾ > 10
```

Control

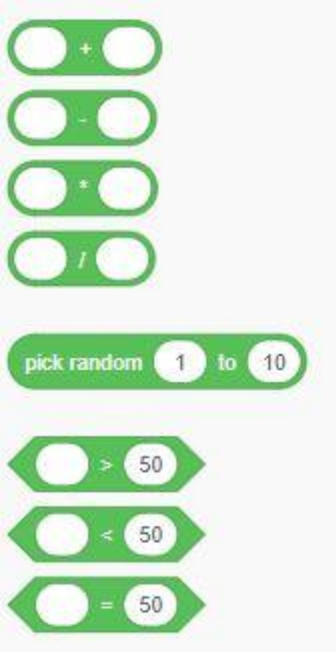
- Control flow and looping sequences of blocks

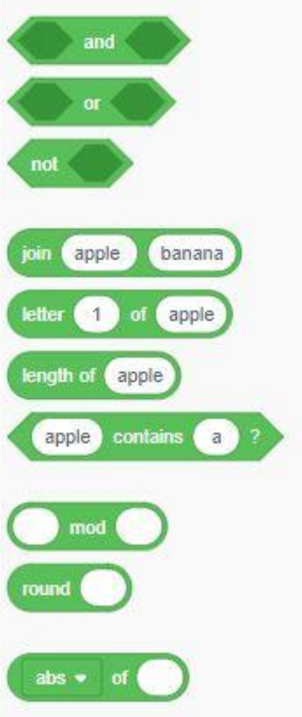




Operators

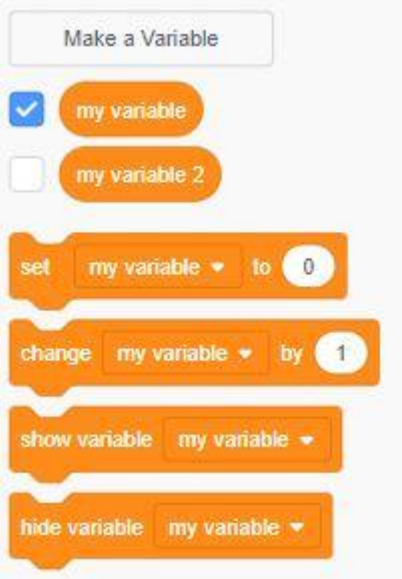
- Mathematical functions and numeric operators
- Boolean and reporter blocks





Variables & Lists

- Reporters to reference the variable/list

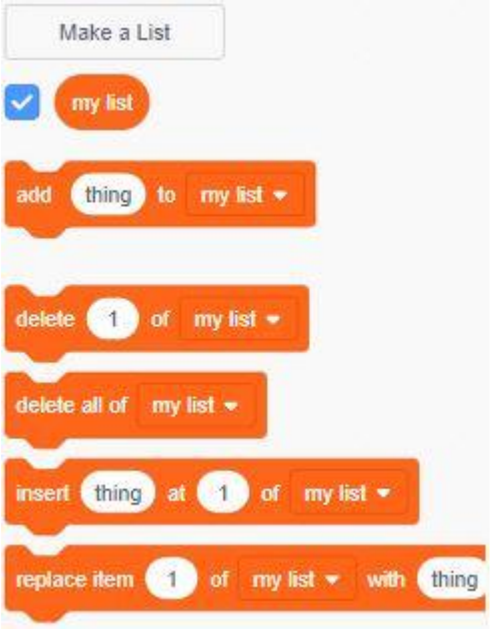


Make a Variable

☑ my variable

□ my variable 2

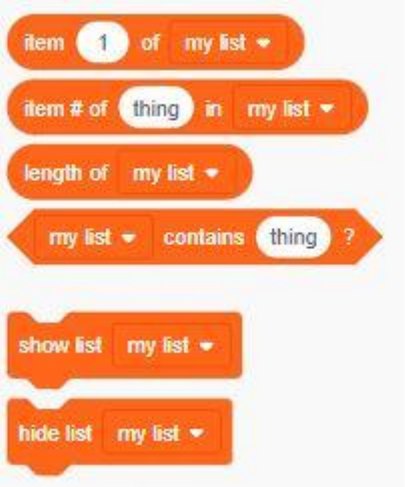
```
set my variable ▾ to 0
change my variable ▾ by 1
show variable my variable ▾
hide variable my variable ▾
```



Make a List

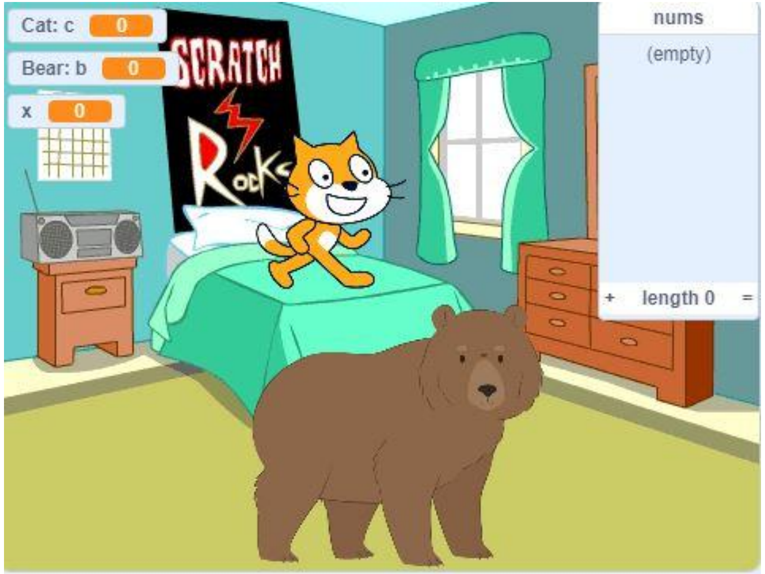
☑ my list

```
add thing to my list ▾
delete 1 of my list ▾
delete all of my list ▾
insert thing at 1 of my list ▾
replace item 1 of my list ▾ with thing
```



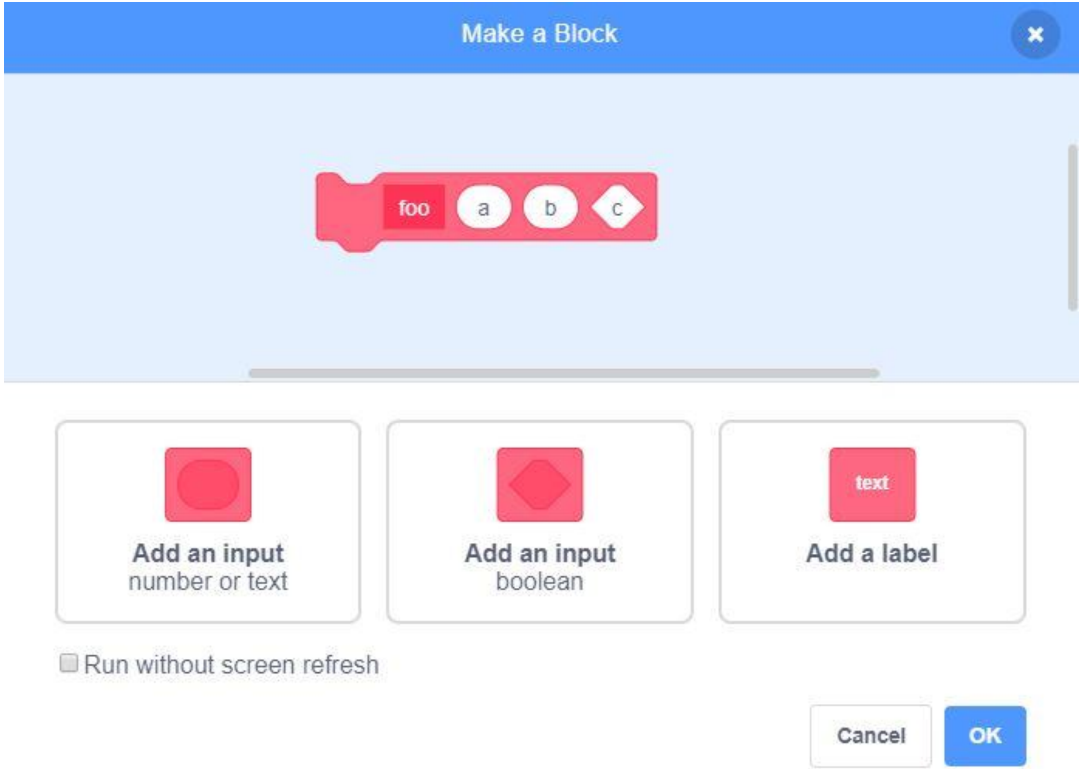
```
item 1 of my list ▾
item # of thing in my list ▾
length of my list ▾
my list ▾ contains thing ?
show list my list ▾
hide list my list ▾
```

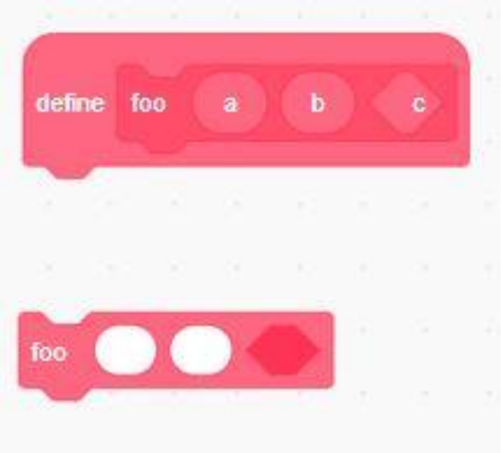
Variables & Lists cont.



- Toggle variables/lists to be displayed
- Cannot reuse variable/list names
- Lists can contain multiple types
- No support for multi-dimensional lists
- Can import text file of comma-separated values into a list
- Fixed list size of 200,000
- Variables can be modified in the GUI with sliders

My blocks



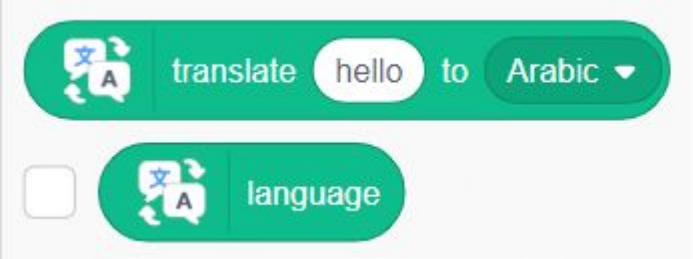


void foo(input a, input b, bool c);

foo(_, _, _);

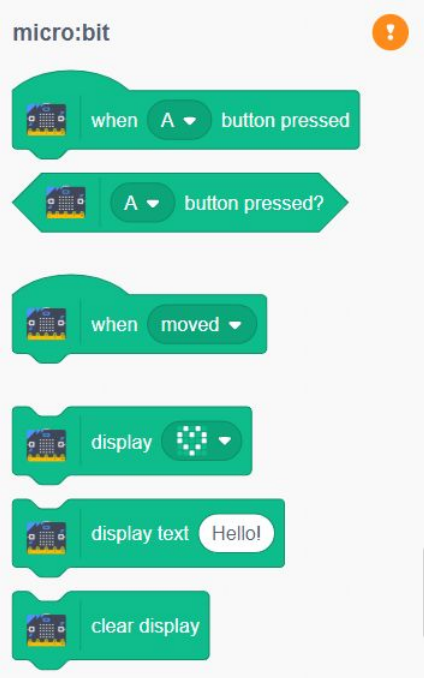
Extensions

- Custom blocks which can be imported



translate hello to Arabic

□ language



micro:bit

when A button pressed

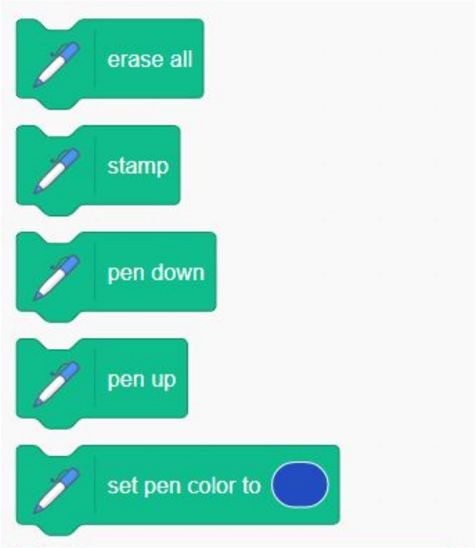
A button pressed?

when moved

display

display text Hello!

clear display



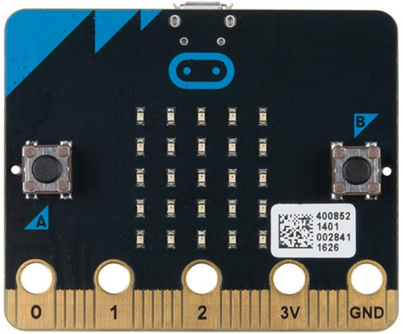
erase all

stamp

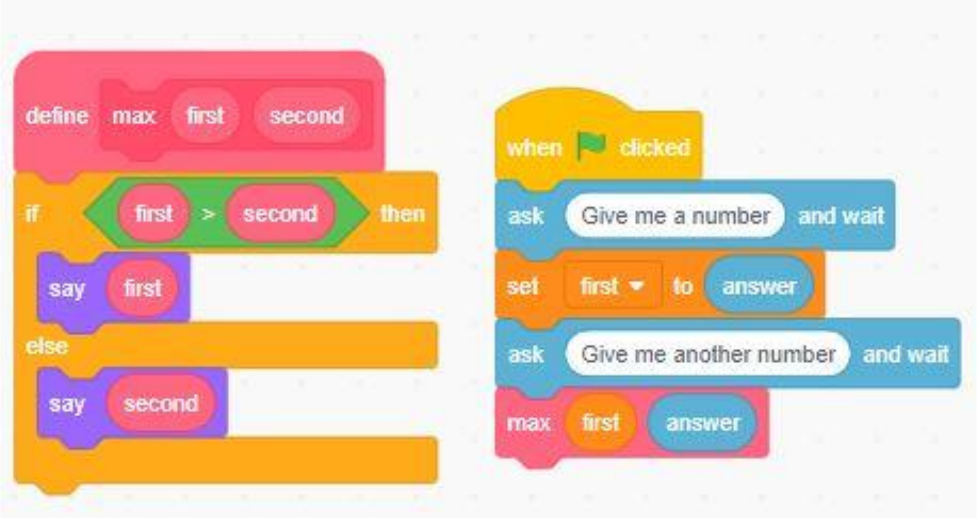
pen down

pen up

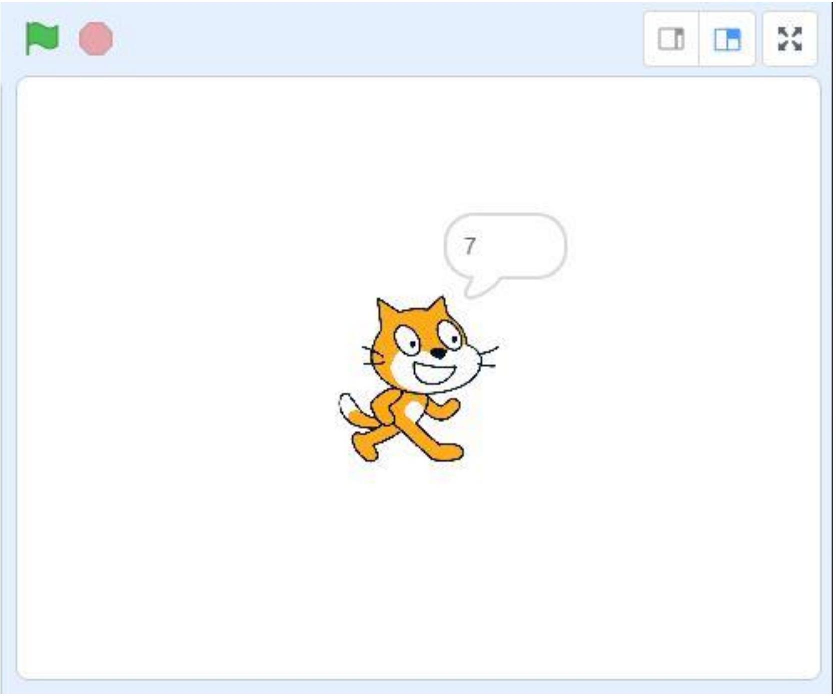
set pen color to



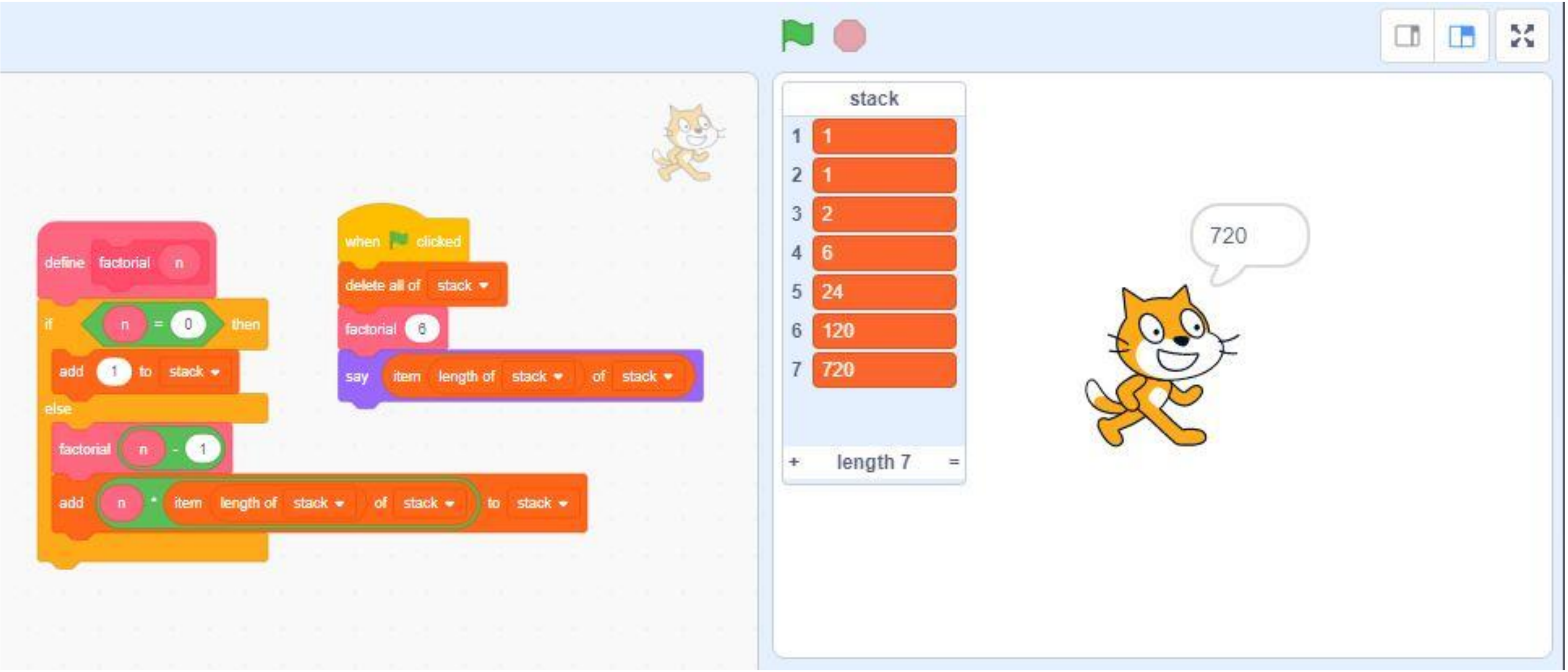
Example 1 - Max



```
define max first second
if < first > second > then
    say first
else
    say second

when ⚑ clicked
ask Give me a number and wait
set first ▾ to answer
ask Give me another number and wait
max first answer
```



Example 2 - Factorial



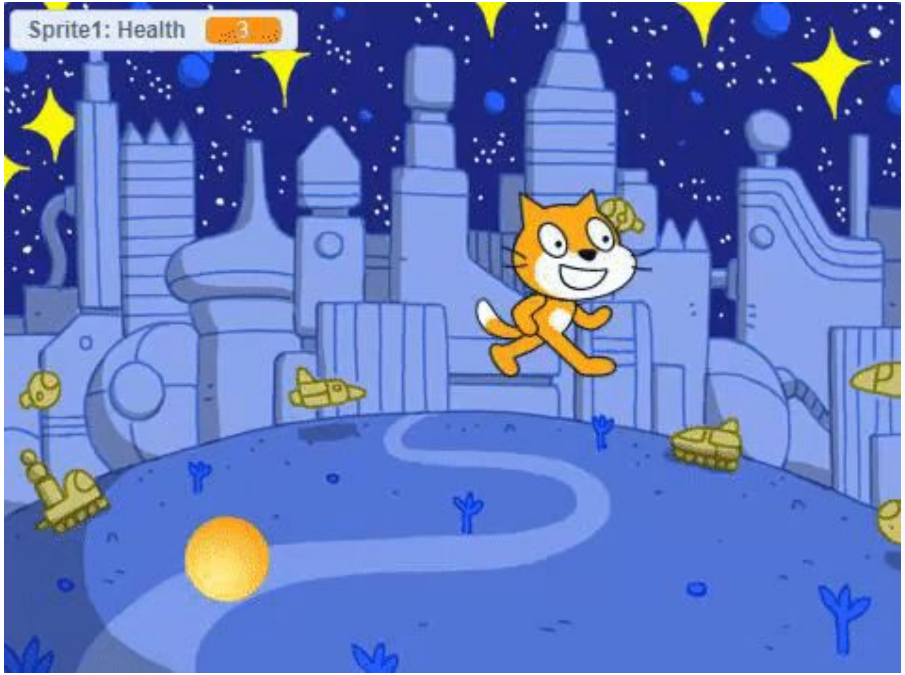
```
define factorial (n)
if <(n) = (0)> then
    add (1) to [stack ▾]
else
    factorial ((n) - (1))
    add ((n) * (item (length of [stack ▾]) of [stack ▾])) to [stack ▾]
```

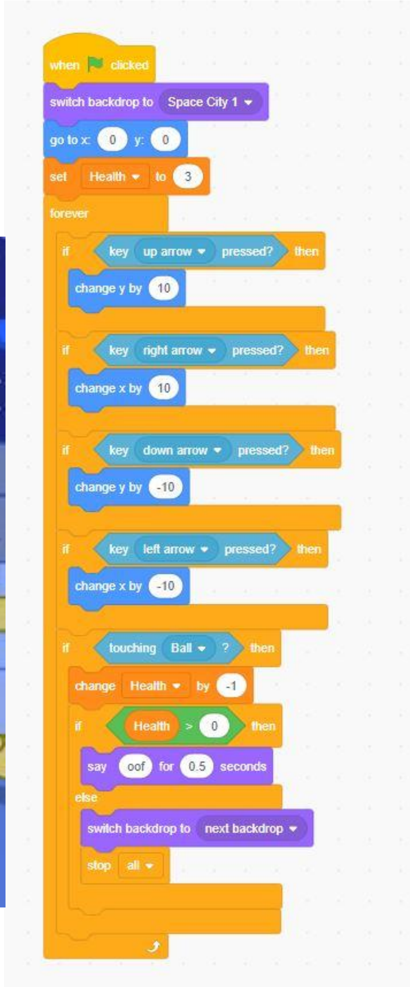
```
when ⚑ clicked
delete all of [stack ▾]
factorial (6)
say (item (length of [stack ▾]) of [stack ▾])
```

```
stack
1  1
2  1
3  2
4  6
5  24
6  120
7  720
+  length 7  =
```

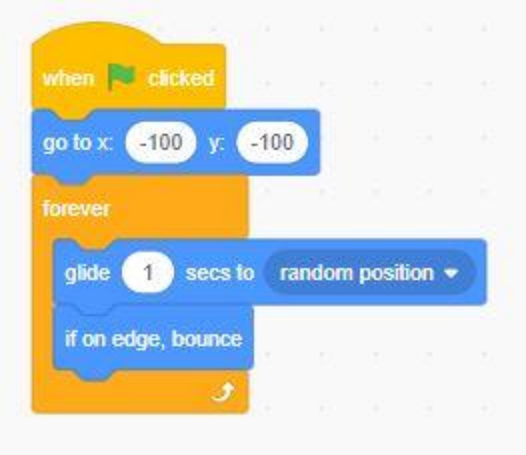
720

Example 3 - Simple game





```
when 🏁 clicked
switch backdrop to Space City 1 ▾
go to x: 0 y: 0
set Health ▾ to 3
forever
    if key up arrow ▾ pressed? then
        change y by 10
    if key right arrow ▾ pressed? then
        change x by 10
    if key down arrow ▾ pressed? then
        change y by -10
    if key left arrow ▾ pressed? then
        change x by -10
    if touching Ball ▾ ? then
        change Health ▾ by -1
        if Health > 0 then
            say oof for 0.5 seconds
        else
            switch backdrop to next backdrop ▾
            stop all ▾
```



```
when 🏁 clicked
go to x: -100 y: -100
forever
    glide 1 secs to random position ▾
    if on edge, bounce
```

Alice

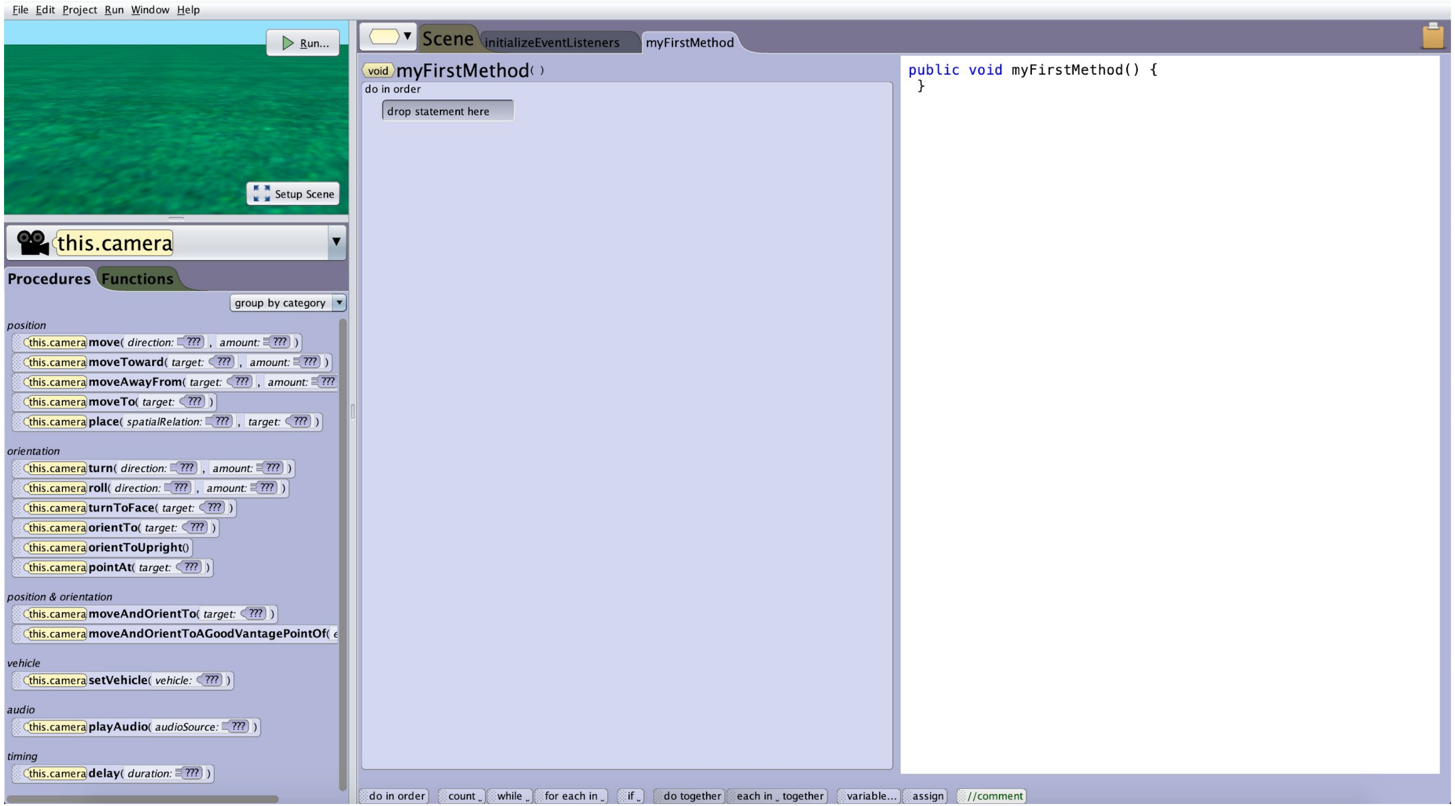
Alice Overview

- What is Alice?
- Alice History
- Alice Features
- Four Core Problems
- Hello World!
- Variables
- Loops
- Recursion
- Threads
- Amusement Park Example

What is Alice?

Alice is a programming environment specifically designed as a teaching/learning tool to enable beginner programmers to create animations and games using 3D worlds and transition into coding in Java

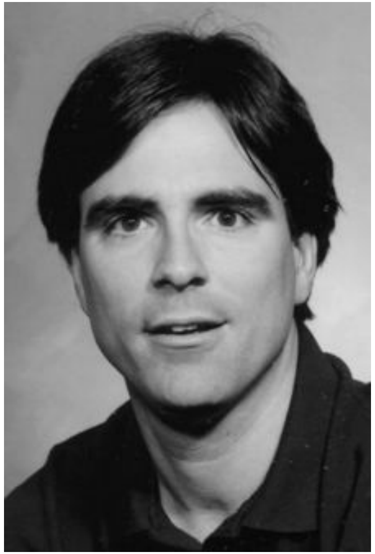
Alice’s Programming Environment

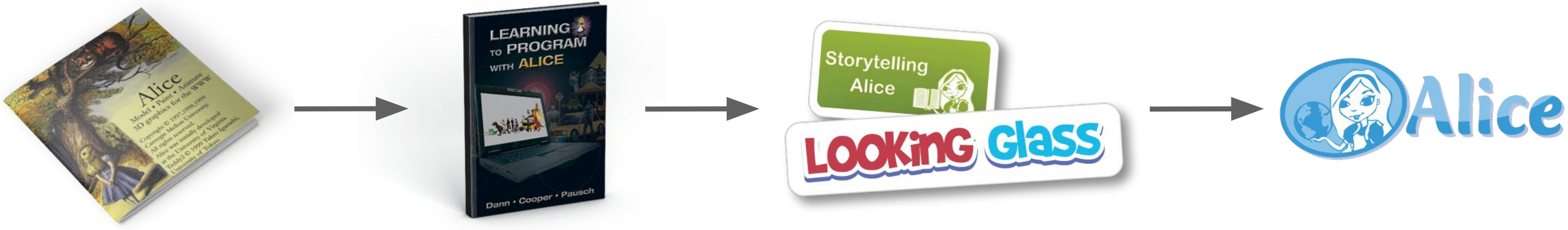

File Edit Project Run Window Help
Run...
Setup Scene
this.camera
Procedures
Functions
group by category
position
this.camera move(direction: ??? , amount: ???)
this.camera moveToward(target: ??? , amount: ???)
this.camera moveAwayFrom(target: ??? , amount: ???
this.camera moveTo(target: ???)
this.camera place(spatialRelation: ??? , target: ???)
orientation
this.camera turn(direction: ??? , amount: ???)
this.camera roll(direction: ??? , amount: ???)
this.camera turnToFace(target: ???)
this.camera orientTo(target: ???)
this.camera orientToUpright()
this.camera pointAt(target: ???)
position & orientation
this.camera moveAndOrientTo(target: ???)
this.camera moveAndOrientToAGoodVantagePointOf(
vehicle
this.camera setVehicle(vehicle: ???)
audio
this.camera playAudio(audioSource: ???)
timing
this.camera delay(duration: ???)
Scene
initializeEventListeners
myFirstMethod
void myFirstMethod( )
do in order
drop statement here
do in order
count
while
for each in
if
do together
each in together
variable...
assign
//comment

```
public void myFirstMethod() {
 }
```

Alice History

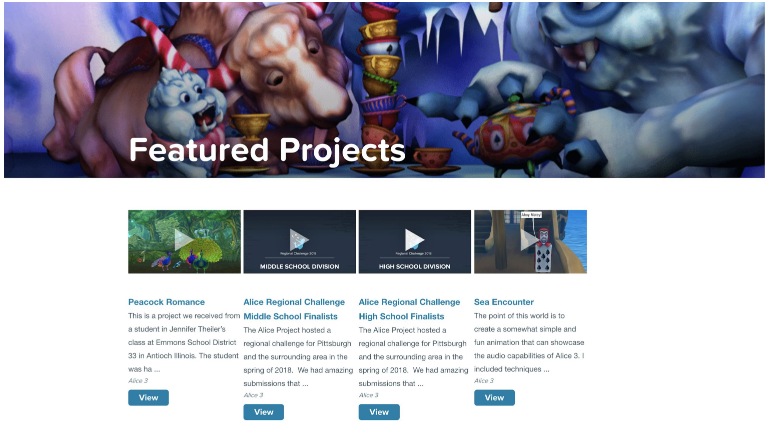
- Created by Randy Pausch at the University of Virginia
- Started as a VR prototyping tool in 1996
- Alice ‘99
- Alice 2
- Storytelling Alice
- Alice 3
- Named ‘Alice’ after Lewis Carroll’s Alice’s Adventures in Wonderland




Alice
Model • Paint • Animate
3D graphics for the WWW
Copyright © 1997,1998,1999
Carnegie Mellon University
All rights reserved.
Alice was initially developed
at the University of Virginia.
Teddy2 © 1999 Takeo Igarashi,
University of Tokyo
LEARNING TO PROGRAM WITH ALICE
Dann • Cooper • Pausch
Storytelling Alice
LOOKinG Glass
Alice

Alice Features

- Featured projects displayed on the main site
- Downloadable environment to build and save projects
- Download project files in .a3p (Alice 3 Project)
- Export a video of running code to YouTube format!



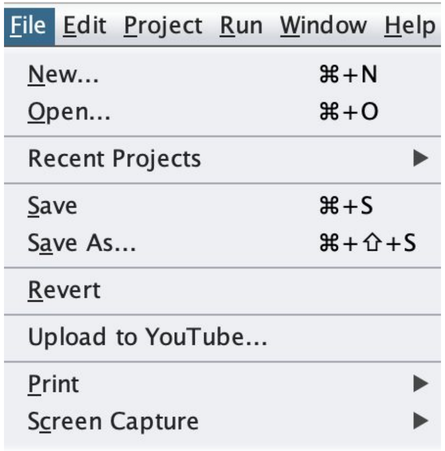
Featured Projects

Peacock Romance
This is a project we received from a student in Jennifer Theiler's class at Emmons School District 33 in Antioch Illinois. The student was ha ...
Alice 3
View

Alice Regional Challenge Middle School Finalists
The Alice Project hosted a regional challenge for Pittsburgh and the surrounding area in the spring of 2018. We had amazing submissions that ...
Alice 3
View

Alice Regional Challenge High School Finalists
The Alice Project hosted a regional challenge for Pittsburgh and the surrounding area in the spring of 2018. We had amazing submissions that ...
Alice 3
View

Sea Encounter
The point of this world is to create a somewhat simple and fun animation that can showcase the audio capabilities of Alice 3. I included techniques ...
Alice 3
View

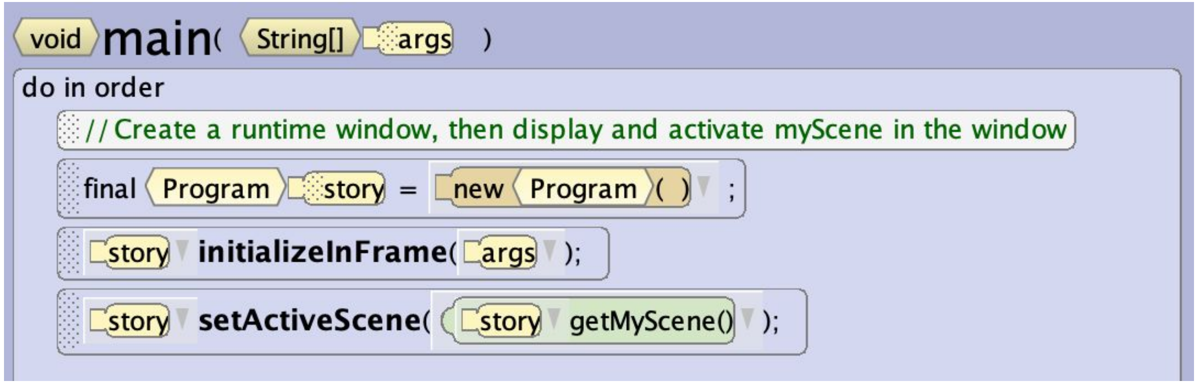


Four Core Problems

- No complex semantics
- No syntax to remember
- Appeal to population unexposed to coding
- Convert code into Java code

Hello World!

Breaking down Hello World!

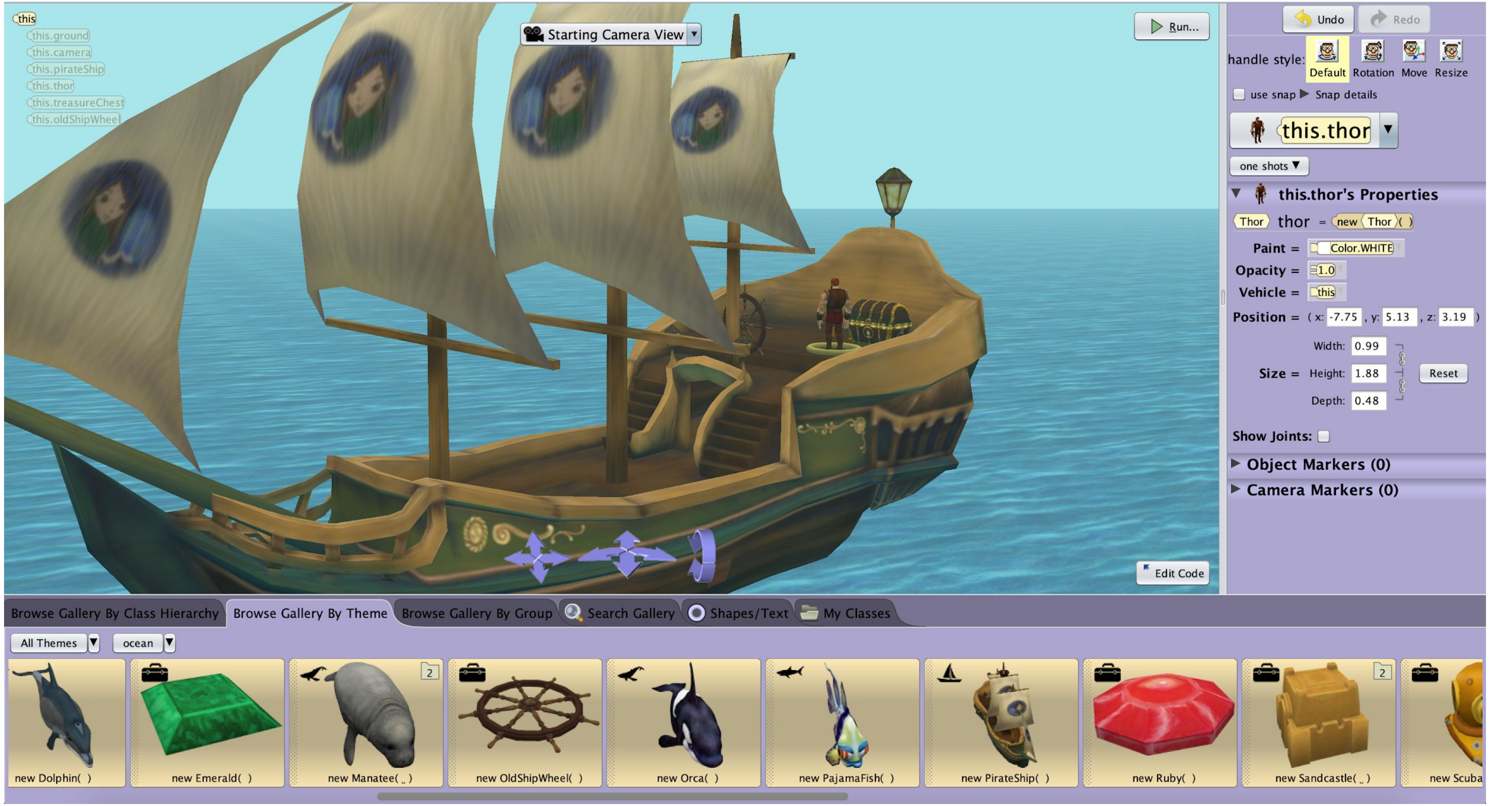


```
void main( String[] args )
do in order
    // Create a runtime window, then display and activate myScene in the window
    final Program story = new Program( ) ;
    story initializeInFrame( args );
    story setActiveScene( story getMyScene() );
```

Default Java main method used in all Alice Projects

```
public static void main( String[] args ) {
    // Create a runtime window, then display and activate myScene in the window
    final Program story = new Program();
    story.initializeInFrame( args );
    story.setActiveScene( story.getMyScene() );
 }
/* End main */
```

Creating the Scene



this
this.ground
this.camera
this.pirateShip
this.thor
this.treasureChest
this.oldShipWheel

this.thor

this.thor's Properties

Thor thor = new Thor( )

Paint = Color.WHITE

Opacity = 1.0

Vehicle = this

Position = (x: -7.75 , y: 5.13 , z: 3.19)

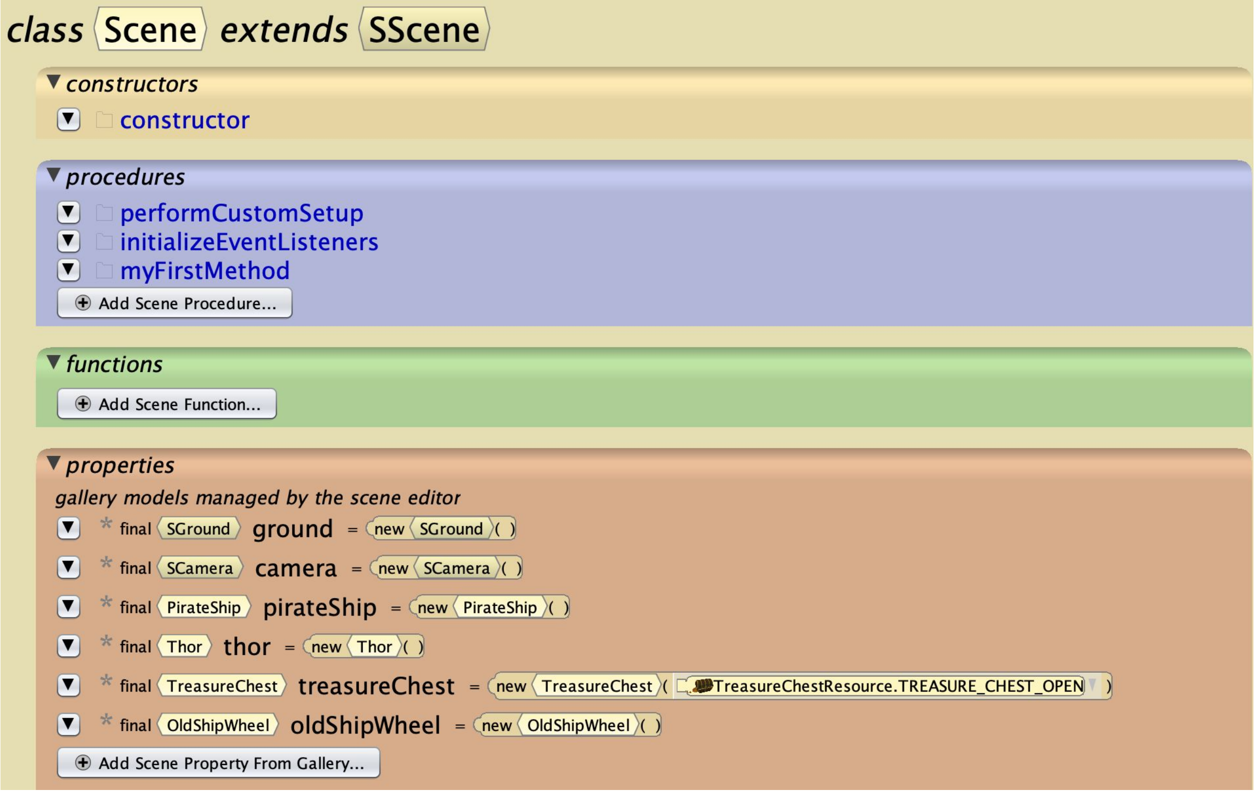
Size = Width: 0.99 Height: 1.88 Depth: 0.48

Show Joints: □

Object Markers (0)

Camera Markers (0)

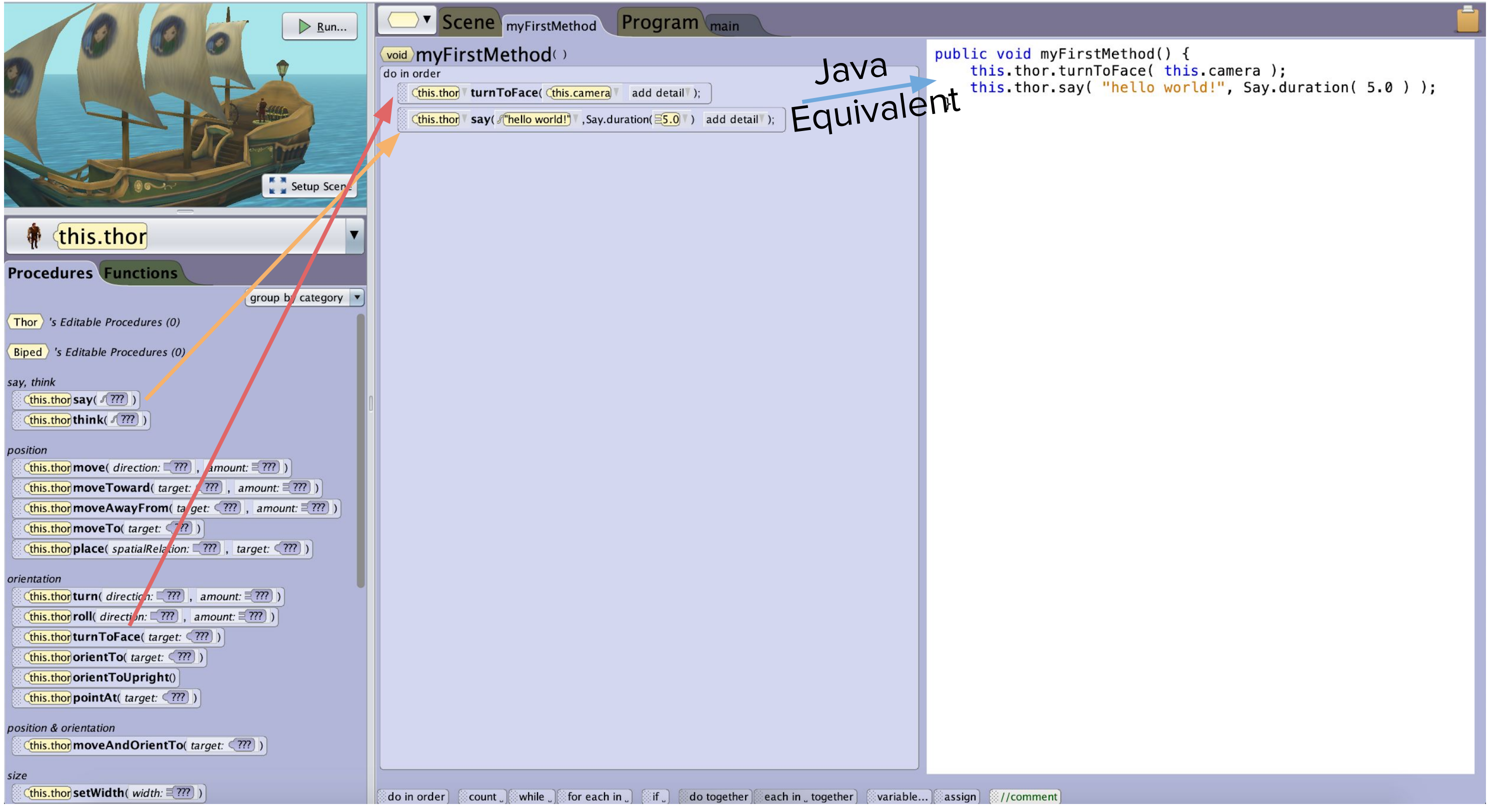
Creating the Scene (Behind the Scenes)



```
class Scene extends SScene
▼ constructors
    constructor
▼ procedures
    performCustomSetup
    initializeEventListeners
    myFirstMethod
    Add Scene Procedure...
▼ functions
    Add Scene Function...
▼ properties
  gallery models managed by the scene editor
    * final SGround ground = new SGround( )
    * final SCamera camera = new SCamera( )
    * final PirateShip pirateShip = new PirateShip( )
    * final Thor thor = new Thor( )
    * final TreasureChest treasureChest = new TreasureChest( TreasureChestResource.TREASURE_CHEST_OPEN )
    * final OldShipWheel oldShipWheel = new OldShipWheel( )
    Add Scene Property From Gallery...
```

Objects displayed on the screen are created in the Scene class as instance fields

Creating the procedure



Run...
Setup Scene
this.thor
Procedures
Functions
group by category
Thor 's Editable Procedures (0)
Biped 's Editable Procedures (0)
say, think
this.thor say(???)
this.thor think(???)
position
this.thor move(direction: ??? , amount: ???)
this.thor moveToward(target: ??? , amount: ???)
this.thor moveAwayFrom(target: ??? , amount: ???)
this.thor moveTo(target: ???)
this.thor place(spatialRelation: ??? , target: ???)
orientation
this.thor turn(direction: ??? , amount: ???)
this.thor roll(direction: ??? , amount: ???)
this.thor turnToFace(target: ???)
this.thor orientTo(target: ???)
this.thor orientToUpright()
this.thor pointAt(target: ???)
position & orientation
this.thor moveAndOrientTo(target: ???)
size
this.thor setWidth(width: ???)
Scene
myFirstMethod
Program
main
do in order
count
while
for each in
if
do together
each in together
variable...
assign
//comment

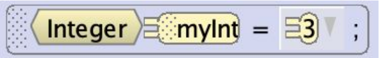
```
void myFirstMethod( )
do in order
    this.thor turnToFace( this.camera  add detail );
    this.thor say( "hello world!" ,Say.duration( 5.0 )  add detail );
```

Java Equivalent

```
public void myFirstMethod() {
    this.thor.turnToFace( this.camera );
    this.thor.say( "hello world!", Say.duration( 5.0 ) );
```

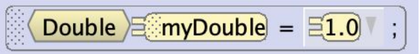
Variables

- Integer



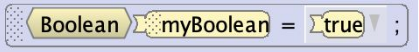
```
Integer myInt = 3 ;
```

- Double



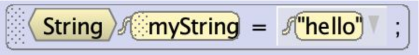
```
Double myDouble = 1.0 ;
```

- Boolean



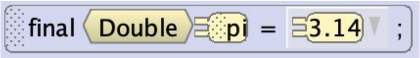
```
Boolean myBoolean = true ;
```

- String



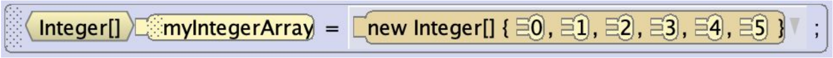
```
String myString = "hello" ;
```

- Constants

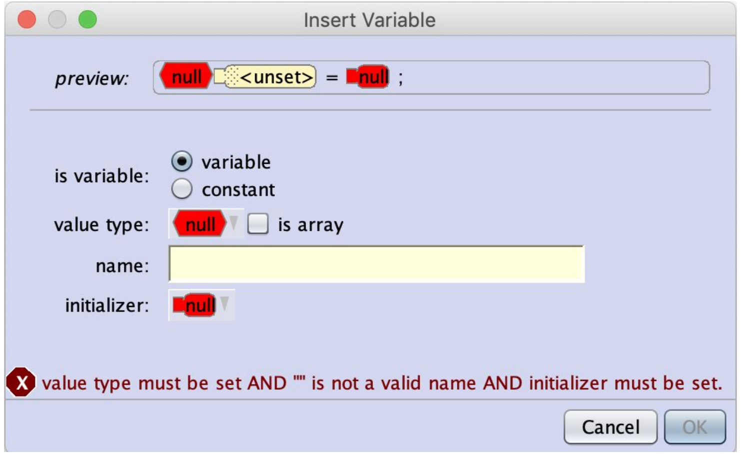


```
final Double pi = 3.14 ;
```

- Arrays



```
Integer[] myIntegerArray = new Integer[] { 0, 1, 2, 3, 4, 5 } ;
```



Insert Variable

preview: null <unset> = null ;

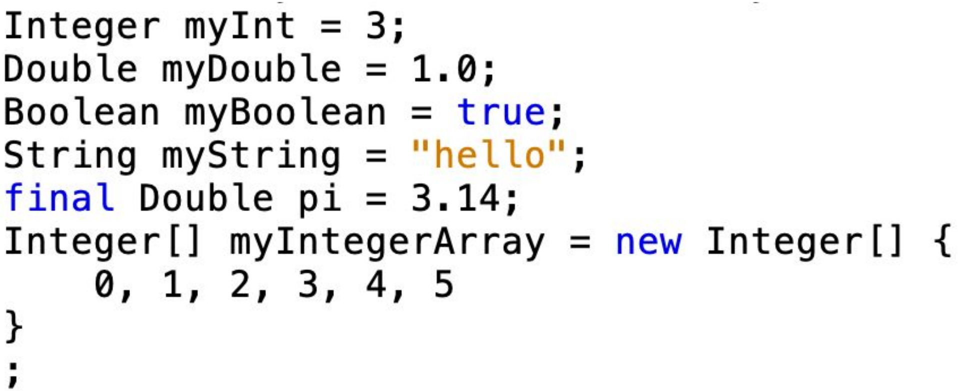
is variable: ● variable ○ constant

value type: null □ is array

name:

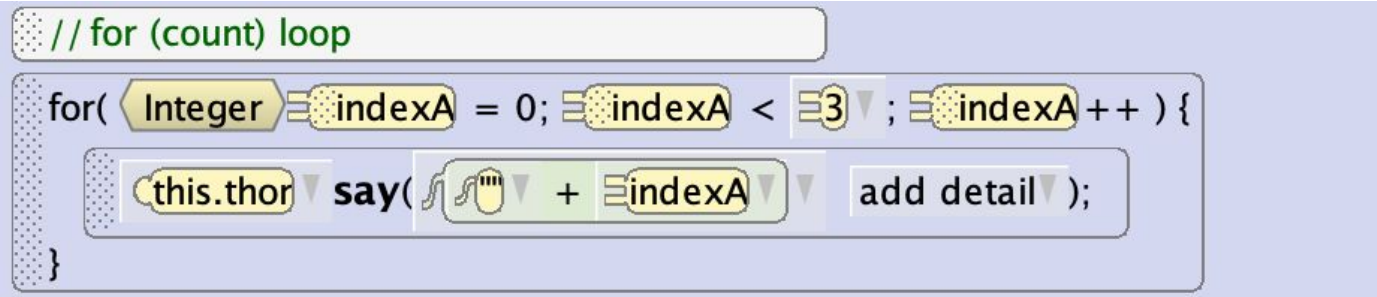
initializer: null

value type must be set AND "" is not a valid name AND initializer must be set.



```
Integer myInt = 3;
Double myDouble = 1.0;
Boolean myBoolean = true;
String myString = "hello";
final Double pi = 3.14;
Integer[] myIntegerArray = new Integer[] {
    0, 1, 2, 3, 4, 5
}
;
```

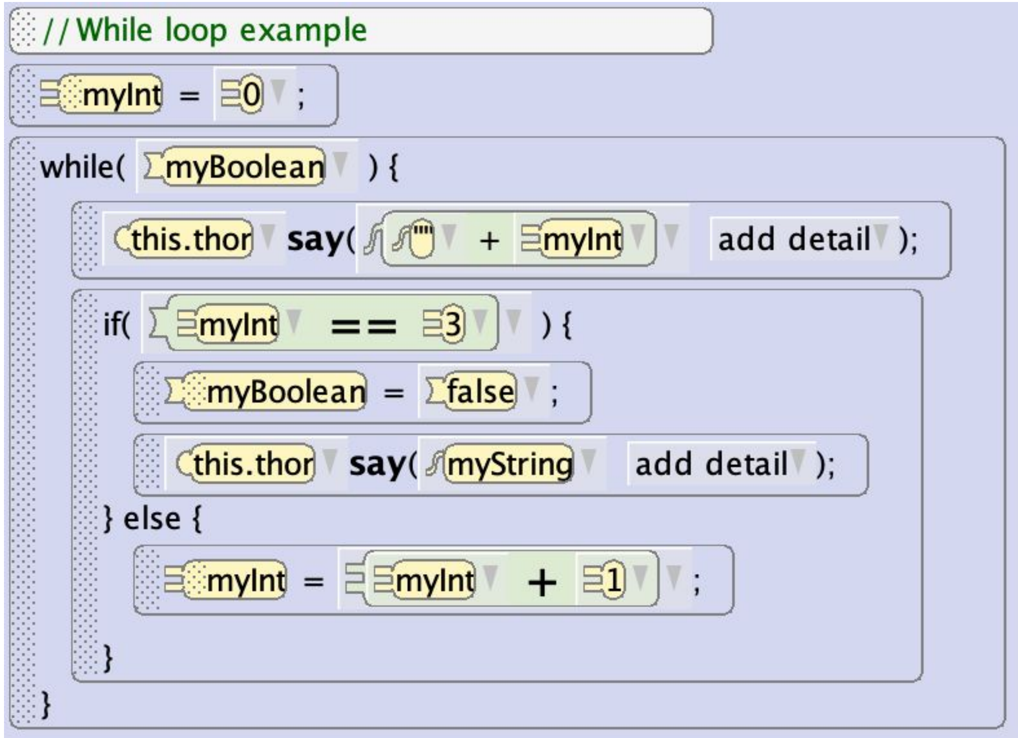
for (count) Loops



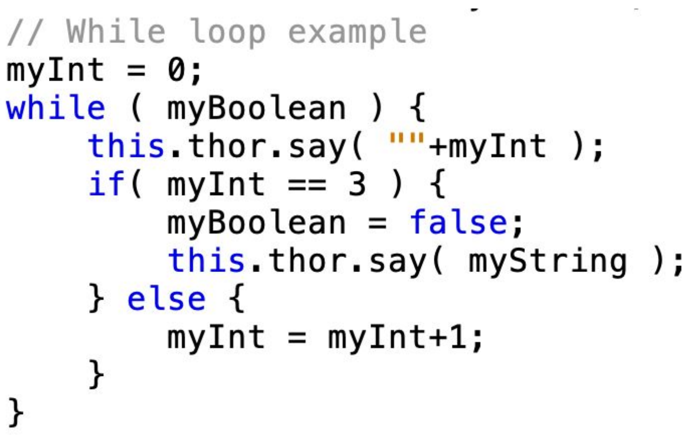
```
// for (count) loop
for( Integer indexA = 0; indexA < 3 ; indexA++ ) {
    this.thor say( "" + indexA add detail );
}
```

```
// for ( count ) loop
for( Integer indexA = 0; indexA < 3; indexA++ ) {
    this.thor.say( ""+indexA );
}
```

while Loops

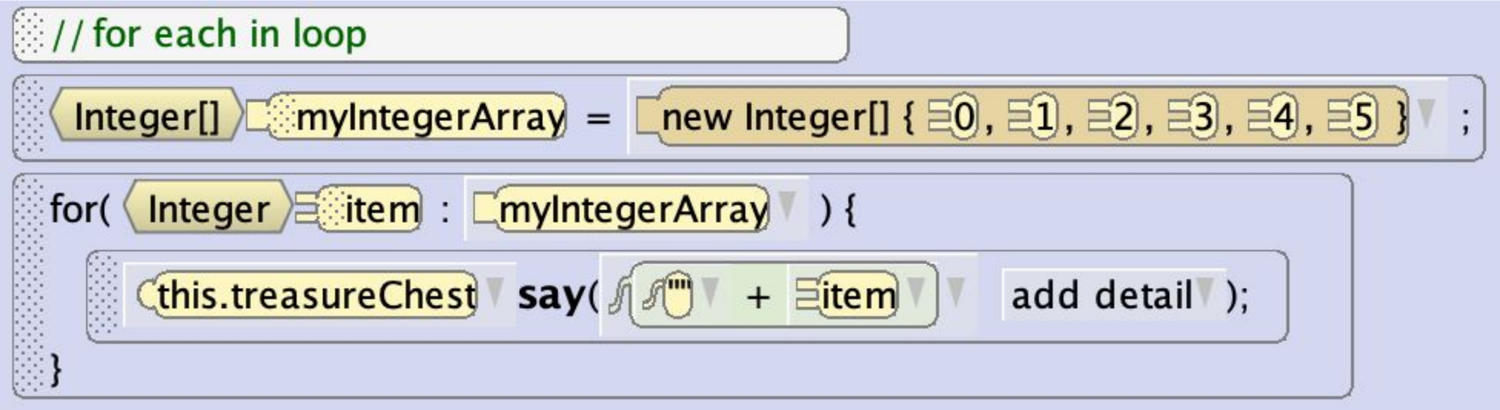


```
// While loop example
myInt = 0 ;
while( myBoolean ) {
    this.thor say( "" + myInt   add detail );
    if( myInt == 3 ) {
        myBoolean = false ;
        this.thor say( myString   add detail );
    } else {
        myInt = myInt + 1 ;
    }
}
```



```
// While loop example
myInt = 0;
while ( myBoolean ) {
    this.thor.say( ""+myInt );
    if( myInt == 3 ) {
        myBoolean = false;
        this.thor.say( myString );
    } else {
        myInt = myInt+1;
    }
}
```

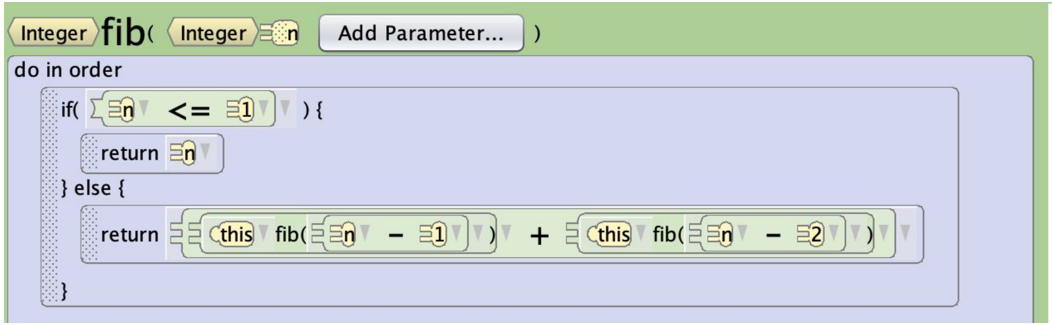
for each in Loops



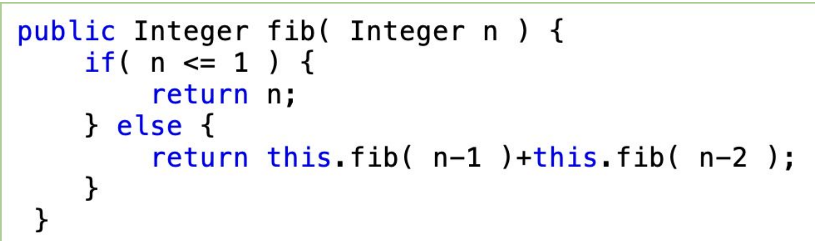
```
// for each in loop
Integer[] myIntegerArray = new Integer[] { 0, 1, 2, 3, 4, 5 } ;
for( Integer item : myIntegerArray ) {
    this.treasureChest say( "" + item   add detail );
}
```

```
// for each in loop
Integer[] myIntegerArray = new Integer[] {
    0, 1, 2, 3, 4, 5
}
;
for( Integer item : myIntegerArray ) { this.treasureChest.say( ""+item ); }
```

Recursion using Fibonacci Example!

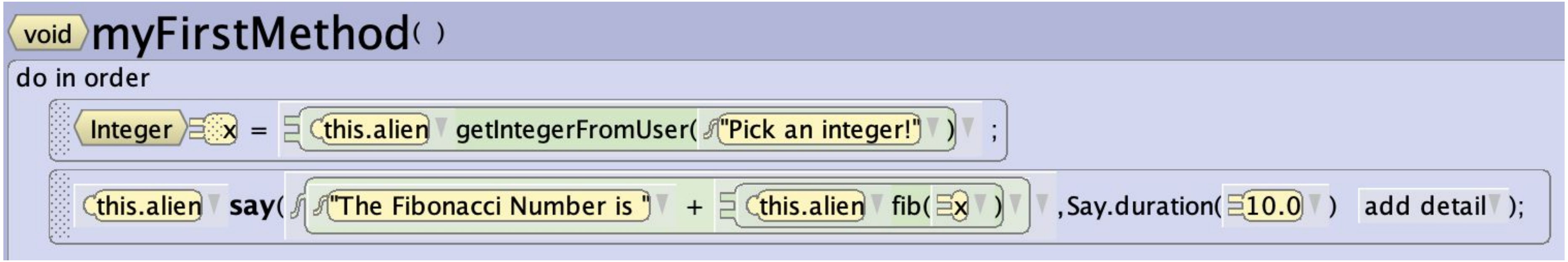


```
Integer fib( Integer n  )
do in order
  if( n <= 1 ) {
    return n
  } else {
    return this fib( n - 1 ) + this fib( n - 2 )
  }
```



```
public Integer fib( Integer n ) {
    if( n <= 1 ) {
        return n;
    } else {
        return this.fib( n-1 )+this.fib( n-2 );
    }
 }
```

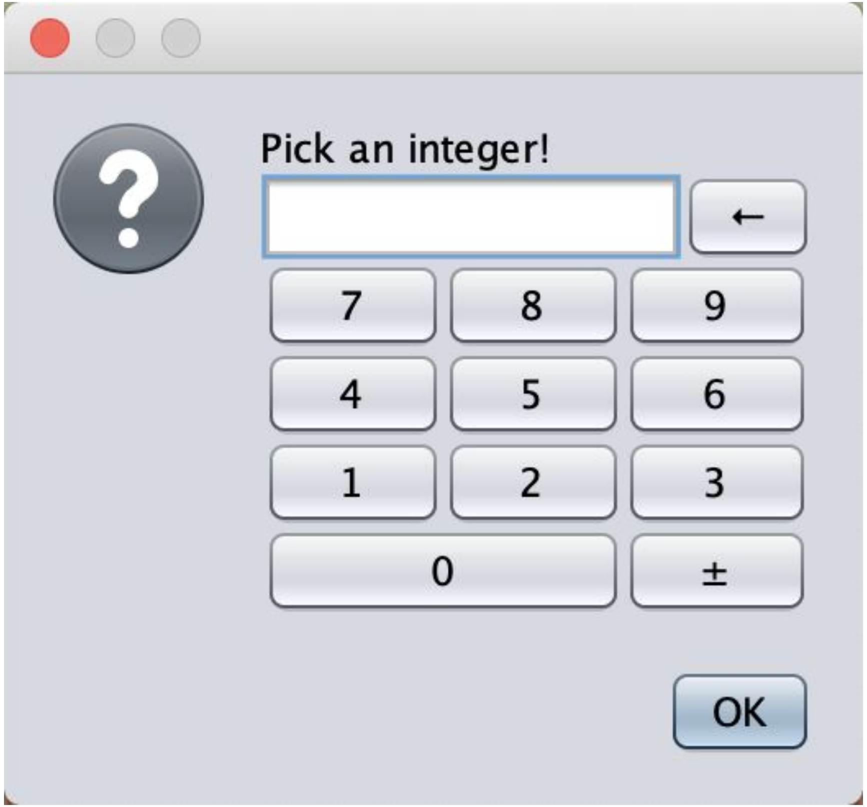
Recursion using Fibonacci Example!



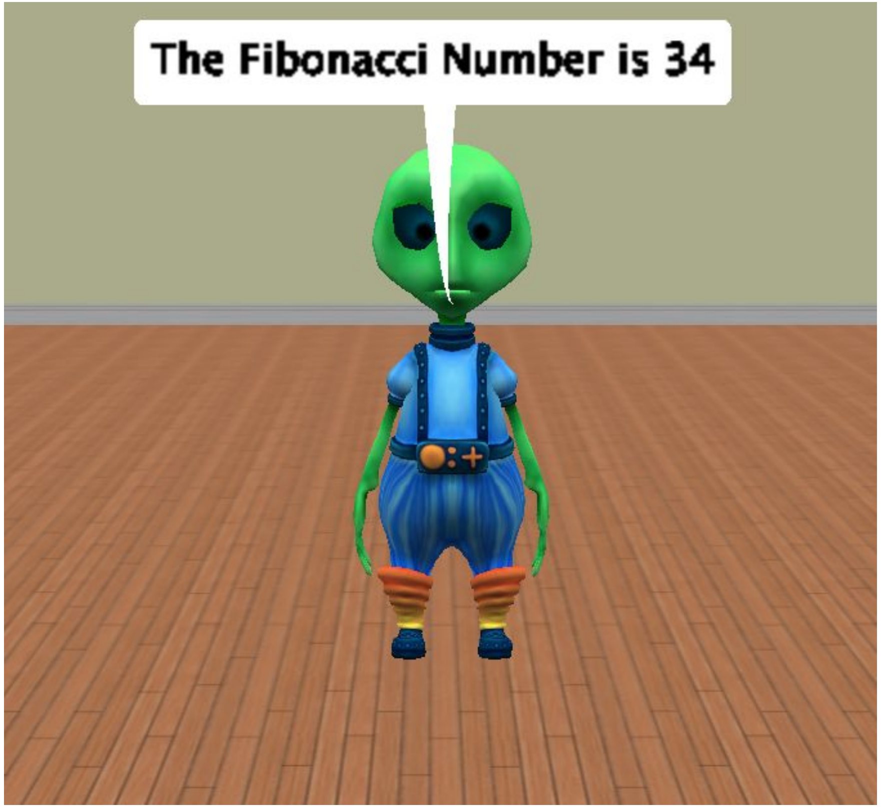
```
void myFirstMethod( )
do in order
    Integer x = this.alien getIntegerFromUser( "Pick an integer!" ) ;
    this.alien say( "The Fibonacci Number is " + this.alien fib( x ) ,Say.duration( 10.0 ) add detail );
```

```
public void myFirstMethod() {
    Integer x = this.alien.getIntegerFromUser( "Pick an integer!" );
    this.alien.say( "The Fibonacci Number is "+this.alien.fib( x ), Say.duration( 10.0 ) );
 }
```

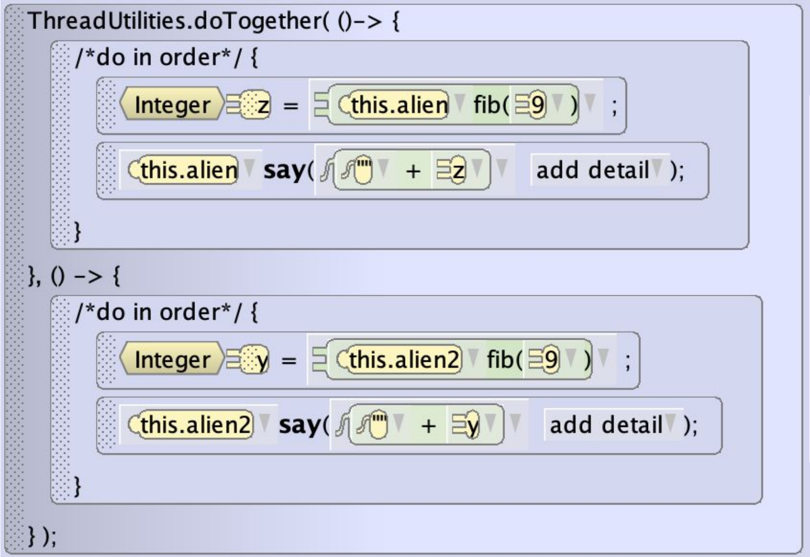
Recursion using Fibonacci Example!



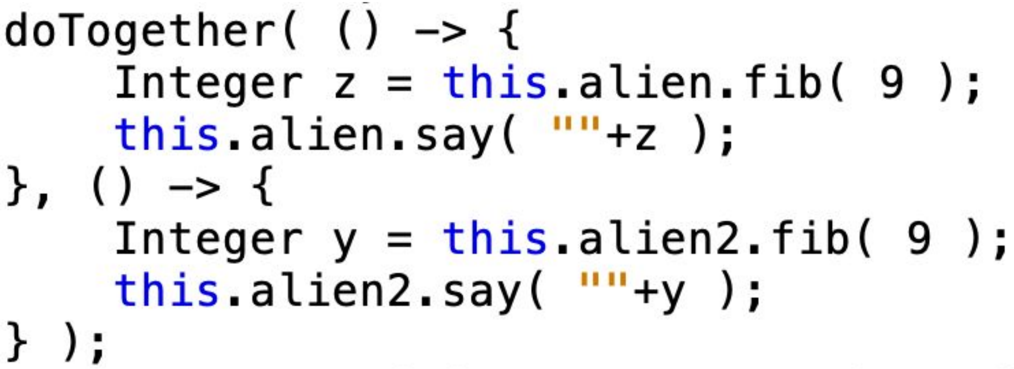
Pick an integer!



Threads - doTogether()

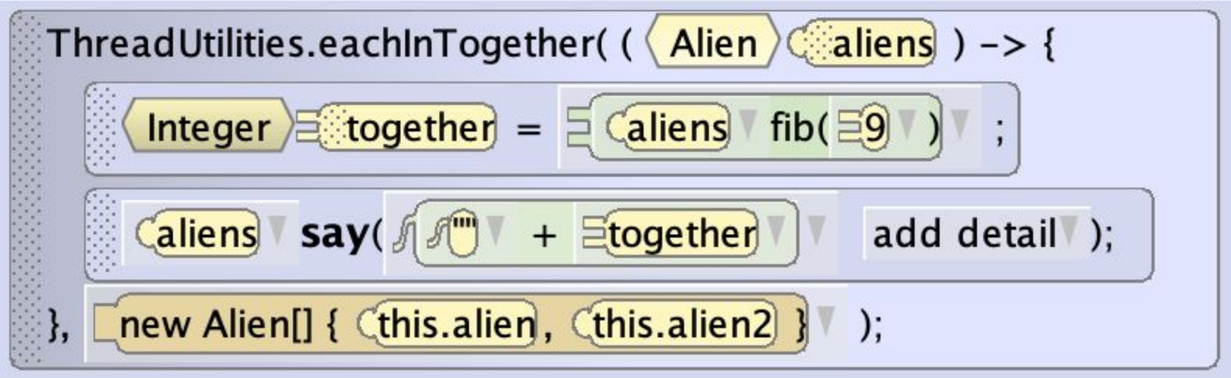


```
ThreadUtilities.doTogether( ()-> {
    /*do in order*/ {
        Integer z = this.alien fib( 9 ) ;
        this.alien say( "" + z  add detail );
    }
}, () -> {
    /*do in order*/ {
        Integer y = this.alien2 fib( 9 ) ;
        this.alien2 say( "" + y  add detail );
    }
} );
```



```
doTogether( () -> {
    Integer z = this.alien.fib( 9 );
    this.alien.say( ""+z );
}, () -> {
    Integer y = this.alien2.fib( 9 );
    this.alien2.say( ""+y );
} );
```

Threads - eachInTogether()



```
ThreadUtilities.eachInTogether( ( Alien aliens ) -> {
    Integer together = aliens fib( 9 ) ;
    aliens say( "" + together   add detail );
}, new Alien[] { this.alien, this.alien2 } );
```

```
eachInTogether( ( Alien aliens ) -> {
    Integer together = aliens.fib( 9 );
    aliens.say( ""+together );
}, this.alien, this.alien2 );
```

Amusement Park Example Demo

Created an example in Alice 2 for a simple amusement park with different features that makes use of all of the concepts covered

Educational Aspects of Scratch and Alice

Educational Aspects Overview

- How Scratch and Alice are used in education
- Impacts on education
- Impacts on computer science
- Benefits and drawbacks of both languages
- Other related languages used in education

How Scratch and Alice are Used in Education

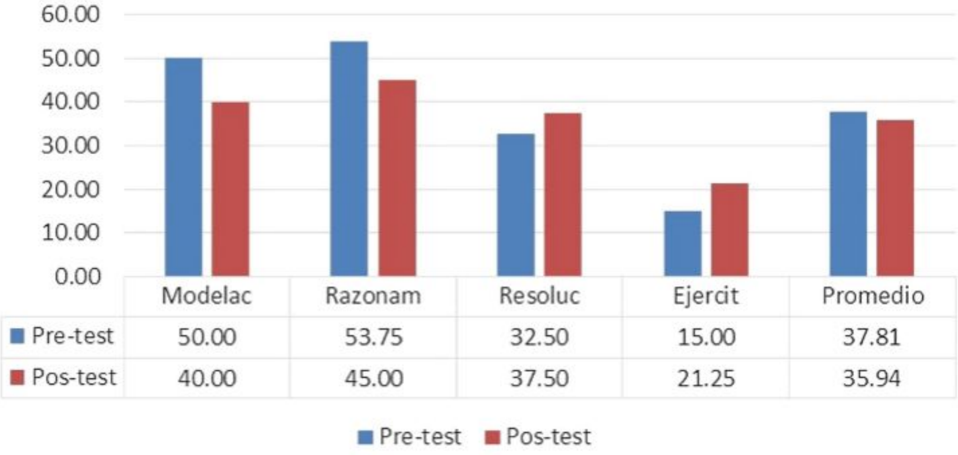
- Computer science concepts and coding
- Computational thinking and problem solving
- Musical live coding
  - Learn basics of music as well as basics of programming
  - University of Massachusetts Lowell dual credit classes
- Storytelling
  - Characters, setting, and plot used as gateways to computer science concepts
  - Kept students engaged when they got frustrated
  - Develops both literary skills and computational skills

Impact on Education

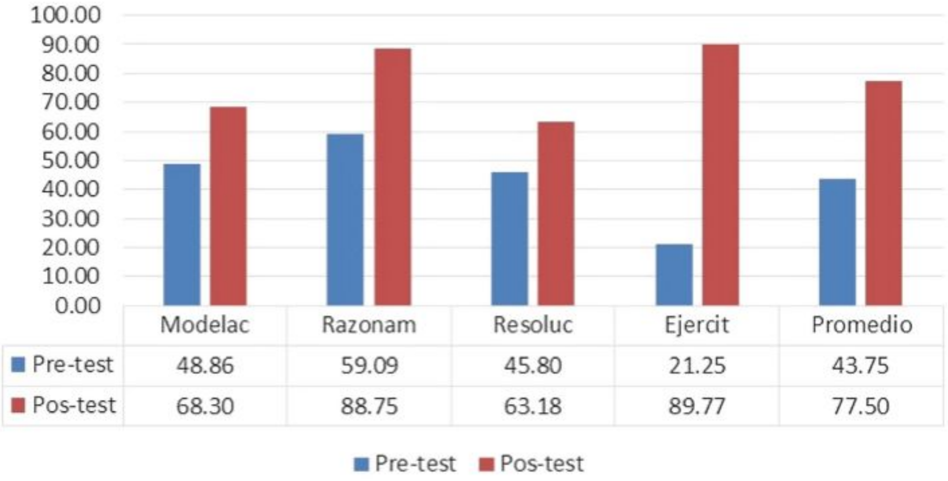
- Creators instead of consumers
- Improves persistence and tinkering skills
- Draws and retains at-risk programming students
    - Makes learning more fun and engaging
- Improves grades for programming courses, even for college students
    - Ithaca College: C → B average, 47% → 88% retention for students with no prior experience
    - Carnegie Mellon: 60% → 84% average in mediated transfer approach
- Storytelling used as a gateway to programming
- Multi-faceted education

Colombia Study

- 46 6th grade students in Colombia
- Control group attended regular math classes
- Experimental group attended modified classes that implemented Scratch and basic programming concepts



	Modelac	Razonam	Resoluc	Ejercit	Promedio
🟦 Pre-test	50.00	53.75	32.50	15.00	37.81
🟥 Pos-test	40.00	45.00	37.50	21.25	35.94



	Modelac	Razonam	Resoluc	Ejercit	Promedio
🟦 Pre-test	48.86	59.09	45.80	21.25	43.75
🟥 Pos-test	68.30	88.75	63.18	89.77	77.50

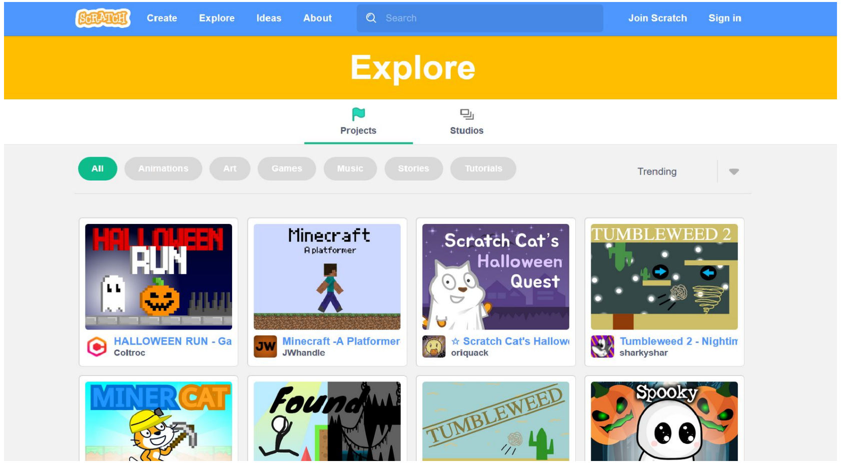
Community Aspects

- Scratch and Alice Communities
  - Projects from around the world
- Community learning encourages students
- More resources for teachers and students
  - Increased exposure to aid if not locally available

11. Jump Through Platforms

Of course, what is missing from a lot of scratch platformers are jump/ drop through platforms.

Have a play with this updated level.. (use the s key too)



Impacts on Computer Science

- More programmers in general
    - Increased retention rates
- Increased diversity in field
    - Female students more likely to join
    - Carnegie Mellon research in middle school
- More knowledgeable about basic computer science concepts
    - Shown by better grades

Benefits and Drawbacks of Scratch

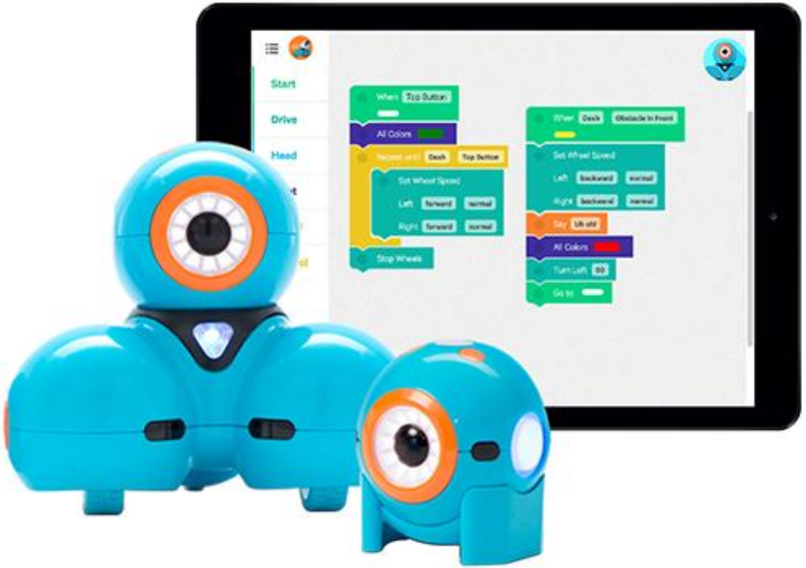
- Benefits
  - Easily followed interface
    - Easy to start without instructor
  - Draws in new programmers at a younger age
  - Increased enthusiasm for learning to code
  - Multi-faceted education
- Drawbacks
  - No converted programming language
    - Doesn’t help learn basic syntax of other languages

Benefits and Drawbacks of Alice

- Benefits
    - Converted into Java
    - Storytelling aspects draw more female coders
        - More likely to continue programming
    - Self expression and sharing experiences
    - Thinking analytically about life experiences
    - Increased retention rates for at-risk students
- Drawbacks
    - Layout and IDE can be confusing
        - Harder to learn without instructor

Related Educational Technologies

- Logo
  - 1960’s programming language for kids developed by Seymour Papert
  - After initial success the language was discontinued in education
- Dot and Dash Robots
  - Introduction to robotics and hardware control
  - Windsor Middle School Makerspace
- Code.org
- TouchDevelop
- Beetle Blocks
  - 3D Design and Fabrication



Thank You!

References

- https://en.wikipedia.org/wiki/Alice_(software)
- https://en.scratch-wiki.info/wiki/Scratch_Wiki
- https://en.wikipedia.org/wiki/Scratch_(programming_language)
- https://en.wikipedia.org/wiki/List_of_educational_programming_languages
- https://en.wikipedia.org/wiki/BASIC
- http://www.hoist-point.com/applesoft_basic_tutorial.htm#def_fn_section
- https://www.sciencedirect.com/science/article/pii/S0360131516300549
- https://https://onlinelibrary.wiley.com/doi/full/10.1111/jcal.12155?casa_token=6v9-fZ71knEAAAAA%3AWAwpHndc96CZQTK7SjRM4vux__vBVriwyLp1rH6CZ0SdRpchxV5-Vczpx0imu9Xvr-OVNfxuXYYMMU8
- link.springer.com/chapter/10.1007/978-3-319-24258-3_2
- https://dl.acm.org/citation.cfm?id=1810611
- https://dl.acm.org/citation.cfm?id=1734384
- http://thales.philos.k.hosei.ac.jp/interaction/cacm50(7)/p58-kelleher.pdf
- https://www.youtube.com/watch?v=ybUGY8-Jn3E