Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective

Mara Saeli, Jacob Perrenet, Wim M.G. Jochems, Bert Zwaneveld
Presented by Martin Quinson

Seminaire IDEES

26 mars 2012

Foreword

About this article

- Published in *Informatics and Education*, 2010. Freely available online.
  `http://www.mii.lt/informatics_in_education/htm/INFE177.htm`
- Authors' institution: *Eindhoven School of Education* (centered on science)
  Three main goal:
  - Teach science to future teachers
  - Conduct research on these themes
  - Support and promote innovations in teaching

About this presentation

- Based on the article, or at least, on my understanding of the article
- You should read the original article for safety
- Please interrupt me if you have questions
- Remember: this is not my own work (it's not even my usual topic ;)

Back to the article

Context and Motivation

- Computers are everywhere, but computer literacy still a rare ability
- Computer Science Education Research (CSER) $\rightsquigarrow$ Improve teaching quality
- Focus restrictions here: teaching programming to 14-18 years old students
- Question: what do the teachers need to know to be good teachers
- Possible Long Term Goals:
  - Write a manual to help teachers to improve
  - Come up with a method to evaluate and diagnose the teachers abilities

What do good teachers know? What's missing to bad ones?

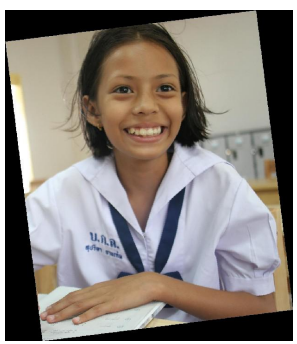
Back to the article

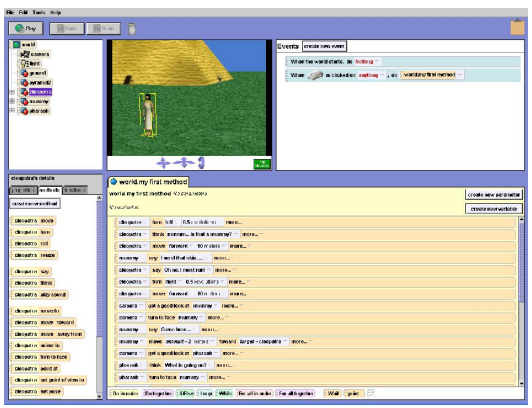
Context and Motivation

- Computers are everywhere, but computer literacy still a rare ability
- Computer Science Education Research (CSER) $\rightsquigarrow$ Improve teaching quality
- Focus restrictions here: teaching programming to 14-18 years old students
- Question: what do the teachers need to know to be good teachers
- Possible Long Term Goals:
  - Write a manual to help teachers to improve
  - Come up with a method to evaluate and diagnose the teachers abilities

What do good teachers know? What's missing to bad ones?

- The students
- Topics Material
- Teaching Methods
- ...





Programming Education

What is Programming (as in Software Development) anyway?

- Wikipedia: Process of writing, testing, debugging/troubleshooting, and maintaining the source of code of computer programs
- Can be broadened: Ability to solve complex problems in a top-down approach
- Hard task: Level of programming understanding tested to be low after 2 years
- But pedagogy helps to some extends (remember of Papert's Logo, 1980)

Which Programming Language?

- Interestingly, this question is completely evacuated by the authors:

> *We will not refer to specific Programming Languages (e.g., Java, Python, etc.), because we consider these as a mean/tool to achieve the teaching of Programming. Secondary school students should be taught programming concepts independent of specific applications and programming languages*
> *(Stephenson et al., 2005; Szlàvi and Zsakò, 2006).*

Pedagogical Content Knowledge

What does PCK mean?

- Concept introduced by Shulman in 1989:

  The ways of representing and formulating the subject that make it comprehensible to others

- Knowing how to program $\not\Rightarrow$ Being able to teach programming (even $\not\Leftarrow$;)
- Combines knowledge of the contents (topics) to knowledge of the pedagogy

What can be PCK?

- Teachers represent and formulate content, so that comprehension can occurs
- Different learners may have different learning styles

$\Rightarrow$ Teachers need a real armamentarium of alternative forms of representation

- It often comes with years of teaching experience (cf. folk pedagogy)

- Rq: PCK can be *general* (domain wide) or *personal* (of a given teacher)
  Personal beliefs on the topic influence the teaching and thus the PCK.

Problem Statement & Overall Methodology

Main Goal: Uncover the PCK of Informatics

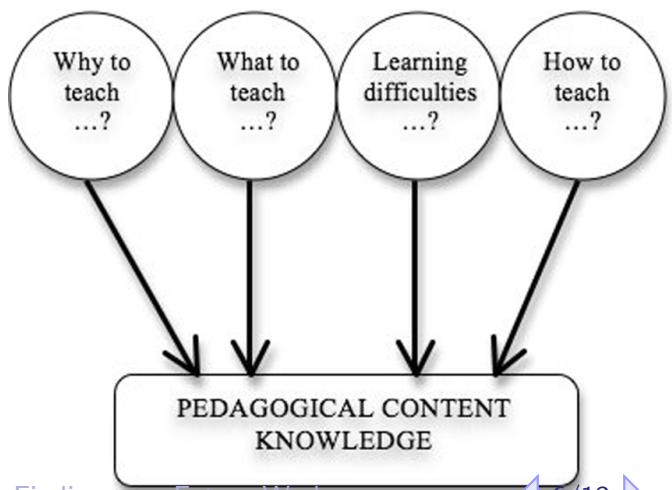
- The authors briefly mention several similar works (see article)
- Portraying the PCK of informatics is still a starting effort

What is out of focus?

- Pedagogy covers a much wider area: student's motivation, gender issues, etc.
- The focus here is on teachers, not on anything else

Methodology

- Use Grossmann's reformulation of PCK (1989, 1990) thru 4 questions:
- Easier way to define PCKs
- Simply answer four questions
- Helps defining the PCK, not discovering them
- Here: lit. review, starting in Google Scholar



So, Why Teach Programming?

- The question should be understood from the teacher perspective
- In particular, we don't speak of student motivation here

To teach powerful problem-solving/design/thinking strategies

- Find a solution, communicate it to the machine, using syntax and grammar
- Ability to split the problem in sub-problems & propose generic solutions
- Decomposition and Generalization reusable elsewhere (maths but also real life)

Improve rigorous communication abilities

- Interacting with an unintelligent machine improves natural language skills
- Rigorous thinking and Debugging reusable elsewhere (maths but also real life)

Technological Mastery & Deepest understanding of scientific disciplines

- Mastering the computer shouldn't be occluded, of course
- New subject mixing Linguistics, Maths, Philosophy, etc. under a new light

What should be Taught?

Programming Knowledge

- Core of programming: problem solving and creating a program as solution
- Two kind of knowledges: Program generation and Program comprehension
  1. Problem analyze; algorithmic solution; translation into program code
  2. Demonstrate an understanding of how a given program works

Program Knowledge (Tactical Programming)

- Data: Variable, data types (procedural); Object, attributes, actions (OOP)
- Instructions: Control Structures, subroutines; Interacting objects and Methods
- Syntax: Language-specific rules how what you can write and how

Modularity and Abstraction (Programming Strategies)

- Primitive expressions: the simplest expressible entities in a language
- Means of combination by which compound elements are built from simpler ones
- Means of abstraction giving name to compounds to manipulate them as units

Program Semantic

- Programs written with different syntax can perform the same semantic task

What are the Learning Difficulties? (1/2)

Programming is a really difficult task

- Correct program often result of an unexpected surprise to beginners

DuBoulay (1989)

- Orientation: Finding out what the benefits to learn to program are;
- Notional Machine: Properties of the machine that one is learning to control; Behavioral relations between notional and physical machine
- Notation: Aspects of the various formal languages (syntax and semantics)
- Structures: Schema or plans to use for small-scale goals (e.g., using a loop)
- Pragmatics: Specify, develop, test and debug programs with available tools

Students' misconceptions may lead to conceptual bugs (Pea 1986)

- Superbug: Some students converse with a computer as if it was a human
- Paralellism bug: several lines active or known by the computer at same time
- Intentionality bug: assume computers can go beyond the information given
- Egocentrism bug: don't do what I say, do what I mean!

What are the Learning Difficulties? (2/2)

Paradigm Change

- When teaching several paradigms together (procedural and OO)
- When switching between paradigms (in particular proc $\rightarrow$ OO, not $\leftarrow$)

Problem Solving Skills

- Limited point of view lead to partial or erroneous solutions
- Difficulties to translate mental intuitions in a communicative way

Issues with the semantic

- Combining program parts into working solution (expr, statmt, ctrl struct ...)
- Debugging: understanding the actual semantic of a given code

Issues with the syntax

- Choosing a programming language with complex syntax $\rightsquigarrow$ extra difficulty

How should the Topic be Taught? (1/2)

- No unique solution, but an armamentarium of tools, methods and tricks
- First, use a *simple* language so that students focus on the semantic

Improve Algorithmic Thinking

- Write simple programs and then combine them (receipts for a cooking machine)
- Give many carefully chosen algorithmic problems (Futschek 2006, Romeike 2008)
- This could even be done w/o computers, but translating remains difficult

Possible Chain for Learning (Linn and Dalbey, 1989)

- From program comprehension to program generation, with three links
- Language features: adapt a provided program to do smth slightly different
- Design skills: how to combine features into programs
  - Templates: stereotypical patterns of code
  - Procedural skills: combine toward new problems (plan, test, reformulate)
- General problem-solving skill: Solve problems using new formal systems
  Such as new a programming language

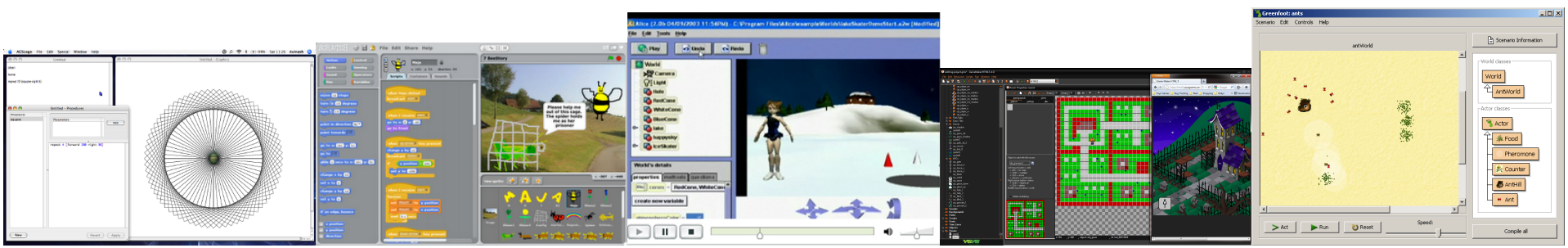
How should the Topic be Taught? (2/2)

Introducing the Notional Machine

- Some model or description of the machine $\rightsquigarrow$ framework for understanding
- Kind of model depends on students' age, background and studies

Engaging the students through adapted environments

- Many environments to introduce programming in active & motivating way
- Logo (problem solving), Scratch (no syntax issue), Alice, Greenfoot, Gamemaker
- Justified by Piaget's model of children's learning:
  Students build their own intellectual structures, if provided with right material
- Teachers provide right support/stimuli/learning material to use with tools
- But no such environment contain all structures & topics needed



Conclusion and Future work

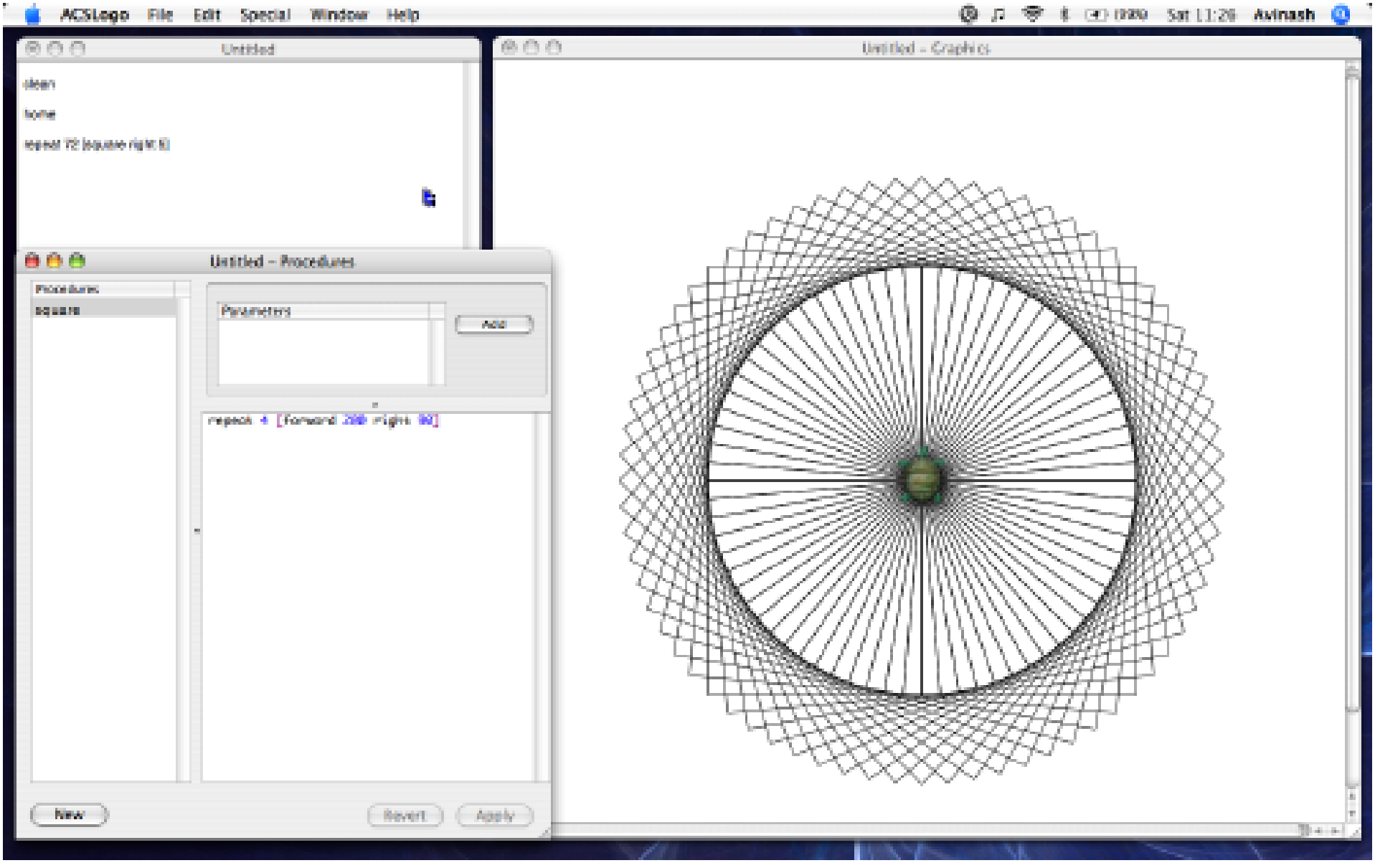
This work is part of a PhD

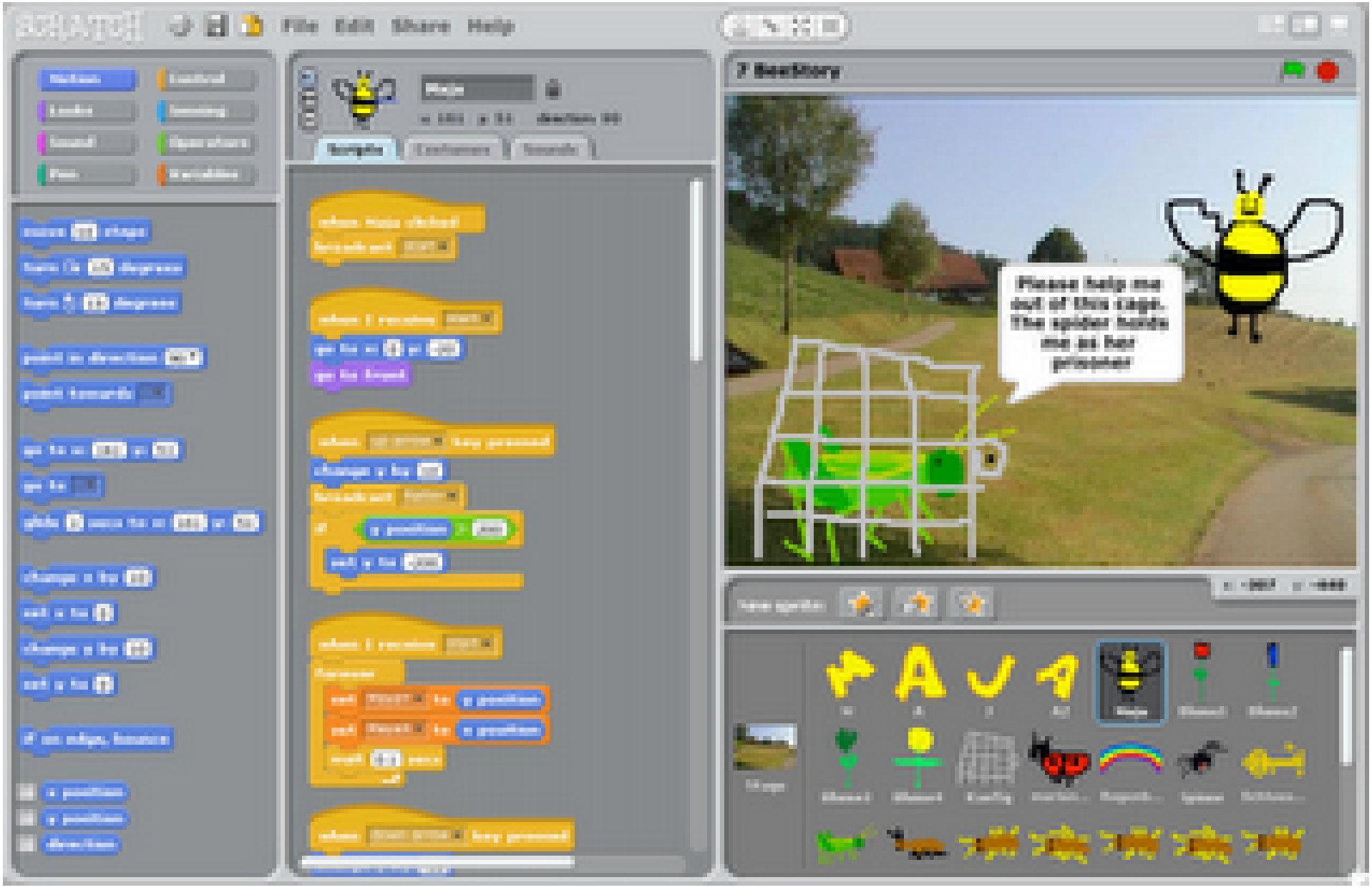
1. Literature review (the journal article)
2. Applies CoRes methodology in workshop with practitioners
   Identify Programming's Big Ideas and their own detailed PCK
3. Assess how a manual could help teachers with low PCK, assess Teachers' PCK

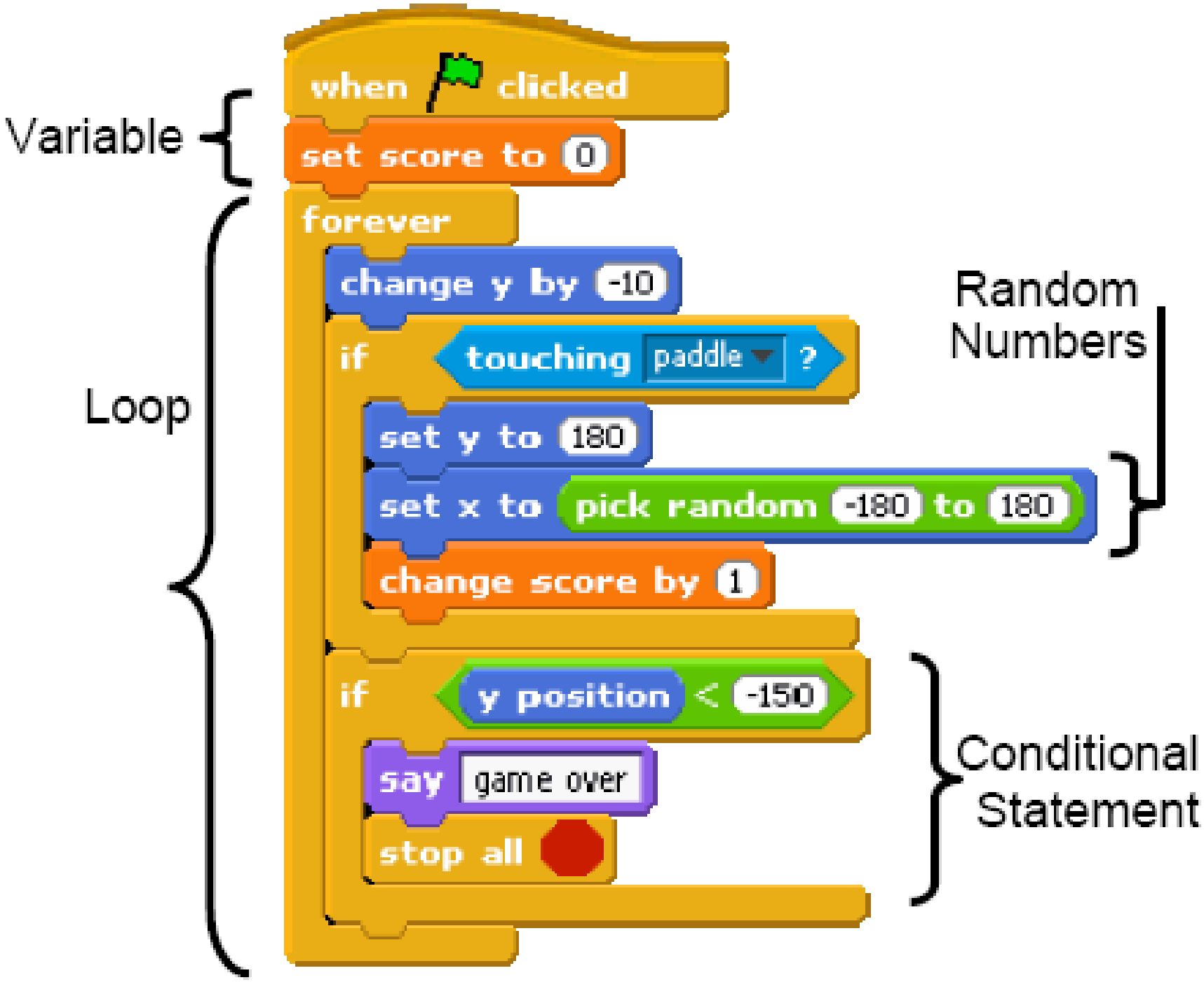
CoRes (Content Representations)

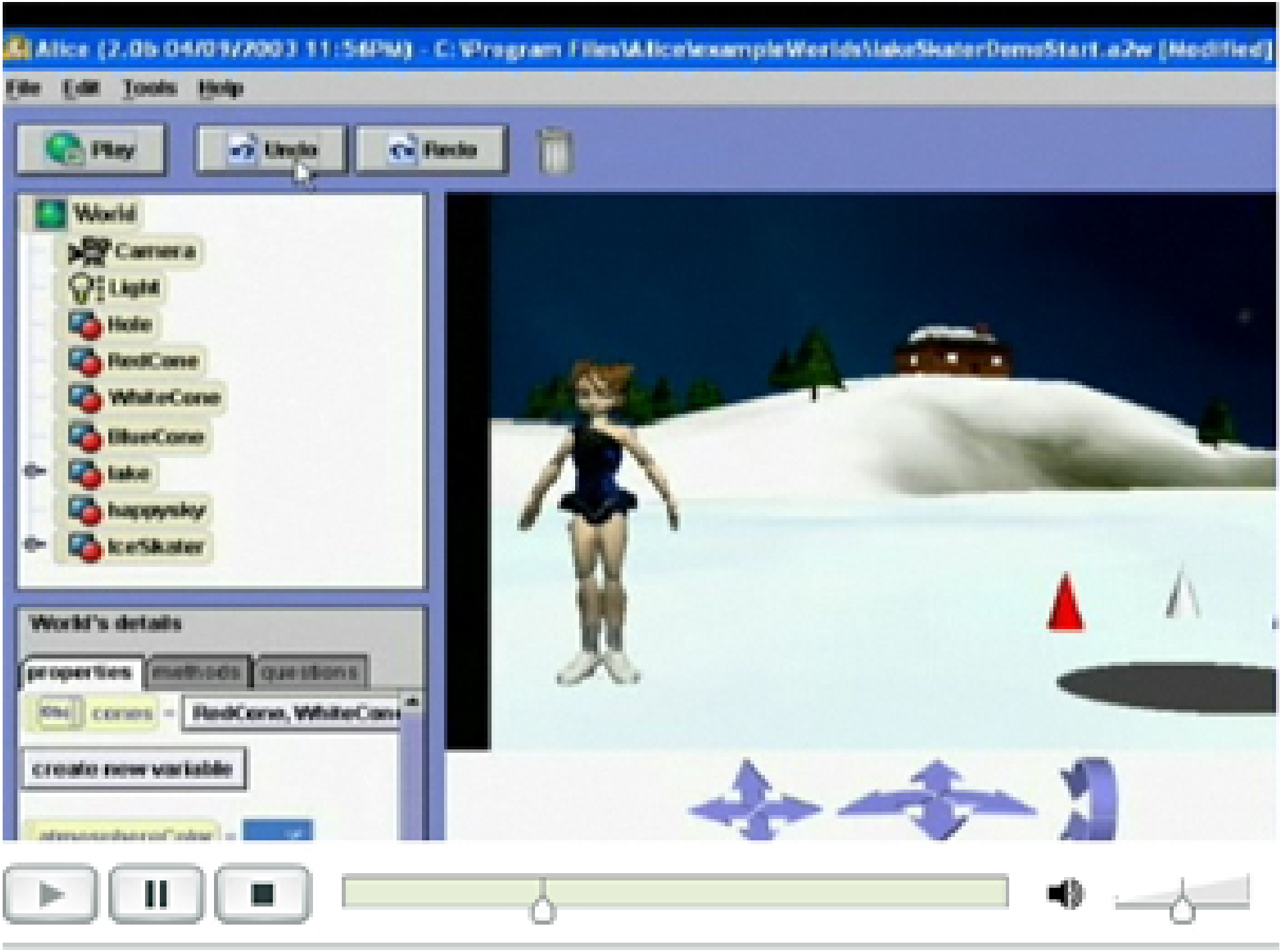
- Research instrument deployed to discover chemistry's PCK in Australia in 2001
- Enumerate the Big Ideas (BI) of the domain, and ask 8 questions for each:
  a. Why is it important for the students to know this BI?
  b. What do you intend the students to learn about this BI?
  c. What else *you* might know about this BI (students don't need to know yet)?
  c. What are difficulties/ limitations connected with the teaching of this Big idea?
  c. What do you think students need to know in order for them to learn this BI?
  c. Specific ways of ascertaining students understanding or confusion around BI?
  d. Which factors influence your teaching of this Big idea?
  d. What are teaching methods (any particular reasons for using these)?

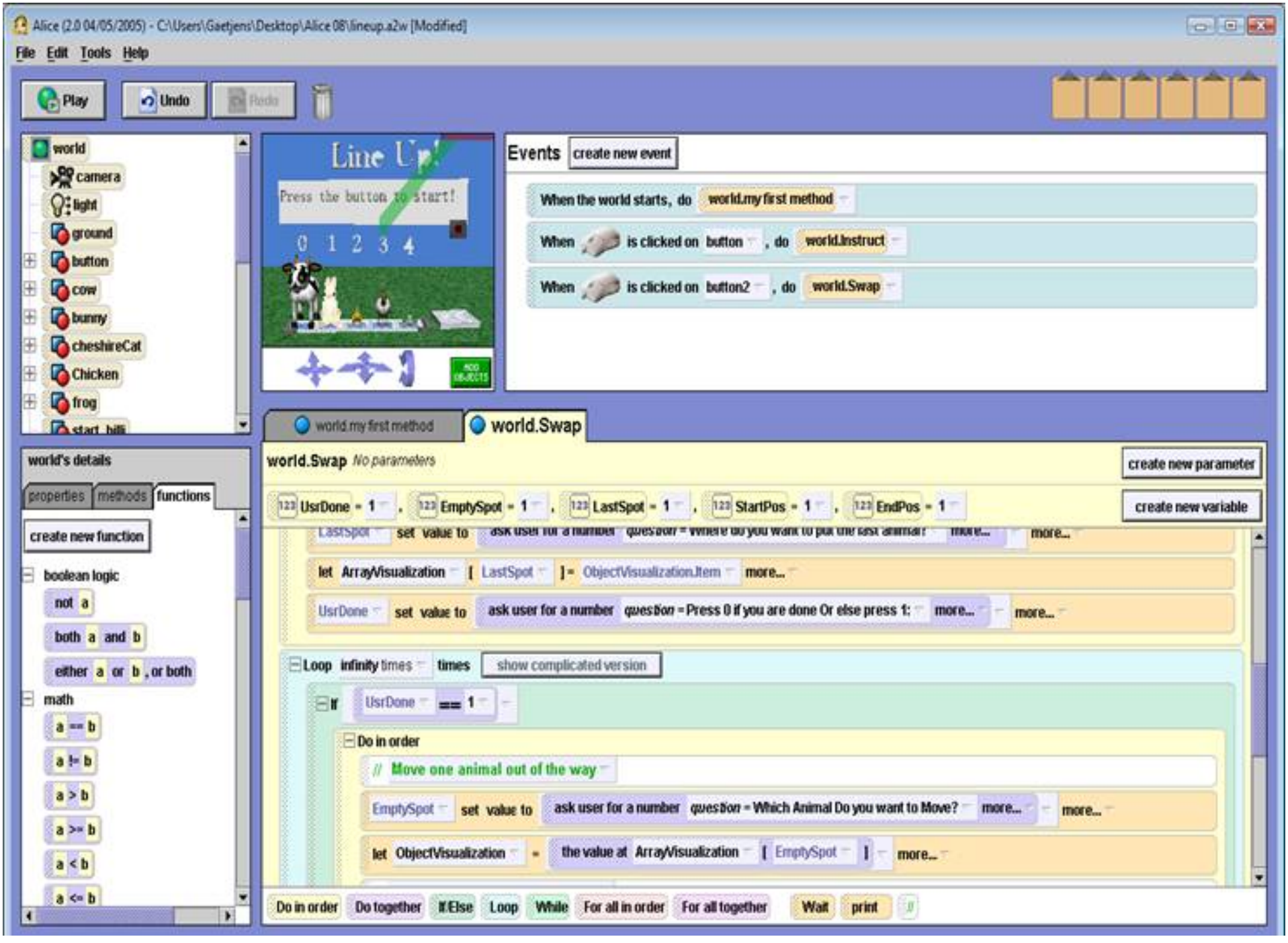
(see Research Report `http://www.tue.nl/fileadmin/content/universiteit/Over_de_universiteit/Eindhoven_School_of_Education/Onderzoek/Projecten_promovendi/Maaza_Saeli_SPJZ_Technical_Report.pdf`)


ACSLogo File Edit Special Window Help
Sat 11:26 Avinash
Untitled
clean
home
repeat 72 [square right 5]
Untitled – Procedures
Procedures
square
Parameters
Add
repeat 4 [forward 200 right 90]
New
Revert
Apply
Untitled – Graphics


File Edit Share Help
Please help me out of this cage. The spider holds me as her prisoner


Variable
Loop
when clicked
set score to 0
forever
change y by -10
if touching paddle ?
set y to 180
set x to pick random -180 to 180
change score by 1
if y position < -150
say game over
stop all
Random Numbers
Conditional Statement


Alice (2.0b 04/09/2003 11:54PM) - C:\Program Files\Alice\exampleWorlds\IceSkaterDemoStart.a2w [Modified]
File Edit Tools Help
Play
Undo
Redo
World
Camera
Light
Hole
RedCone
WhiteCone
BlueCone
lake
happysky
IceSkater
World's details
properties
methods
questions
RedCone, WhiteCone
create new variable

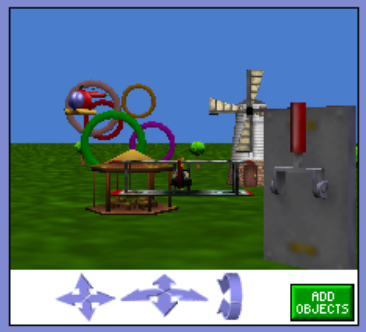

Alice (2.0 04/05/2005) - C:\Users\Gaetjens\Desktop\Alice 08\lineup.a2w [Modified]
File Edit Tools Help
Play
Undo
Redo
world
camera
light
ground
button
cow
bunny
cheshireCat
Chicken
frog
Line Up!
Press the button to start!
0 1 2 3 4
Events
create new event
When the world starts, do world.my first method
When is clicked on button, do world.Instruct
When is clicked on button2, do world.Swap
world.my first method
world.Swap
world's details
properties
methods
functions
create new function
boolean logic
not a
both a and b
either a or b, or both
math
a == b
a != b
a > b
a >= b
a < b
a <= b
world.Swap No parameters
create new parameter
UsrDone = 1
EmptySpot = 1
LastSpot = 1
StartPos = 1
EndPos = 1
create new variable
LastSpot set value to ask user for a number question = Where do you want to put the last animal? more... more...
let ArrayVisualization [LastSpot] = ObjectVisualization.Item more...
UsrDone set value to ask user for a number question = Press 0 if you are done Or else press 1: more... more...
Loop infinity times times
show complicated version
If UsrDone == 1
Do in order
// Move one animal out of the way
EmptySpot set value to ask user for a number question = Which Animal Do you want to Move? more... more...
let ObjectVisualization = the value at ArrayVisualization [EmptySpot] more...
Do in order
Do together
If/Else
Loop
While
For all in order
For all together
Wait
print

Alice

Alice (2.0.7) - /Volumes/Alice-2.0.7_for_OSX10.4/Alice.app/Contents/Required/exampleWorlds/flightSimulator.a2w

File Edit Tools Help

Play | Undo | Redo



- World
  - Camera
  - Light
  - Ground
  - Biplane
  - windmill
  - Gazebo
  - Helicopter
  - TallTree
  - TallTree2

World's details

properties | methods | functions

- WindMillIsOn = true
- turnSpeed = 0.12
- Rings = Torus, Torus2, Toru
- RingsAcquired = 0
- RingsToWinPrize = 5
- rollLeft = true

create new variable

- atmosphereColor =
- ambientLightColor =
- ambientLightBrightness = 1
- fogStyle = no fog
- fogDensity = 0.1
- fogNearDistance = 1 meter
- fogFarDistance = 256 meters

Seldom Used Properties
Sounds
Texture Maps

Events create new event

When the world starts, do StartScreen set opacity to 0.6 (60%) more...

When  is clicked on anything, do World.BeginFlying

While World.WindMillIsOn is true
Begin: Nothing
During: windmill.Blades roll at speed left *speed* = 0.25 revolutions per second more...
End: Nothing

World.TestForCrash

World.TestForCrash *No parameters*

BlinkDuration = 0.05

If Biplane.Propeller distance above Ground more... < 0

Do together

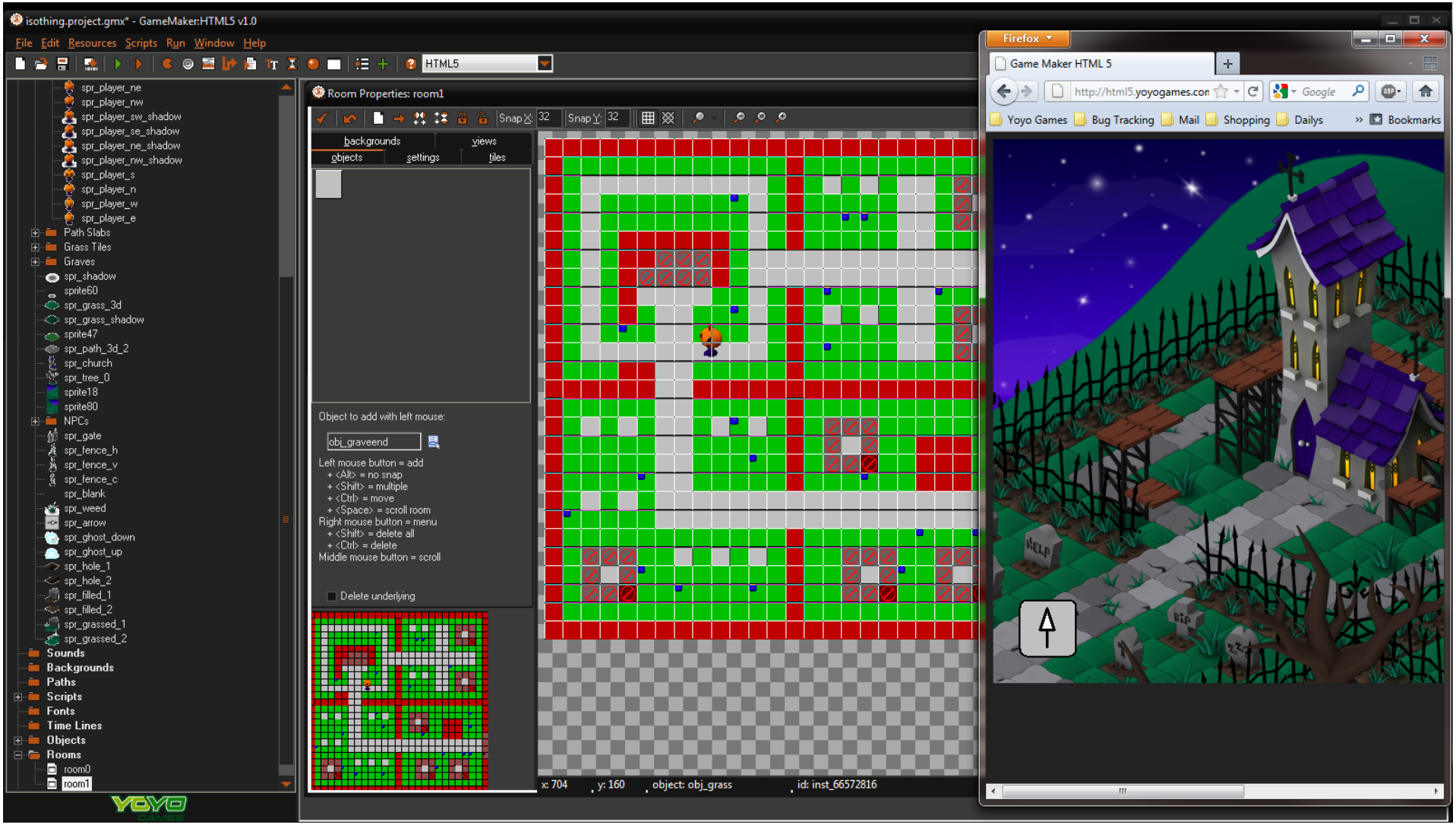
Biplane play sound World.itburns (0:02.257) more...

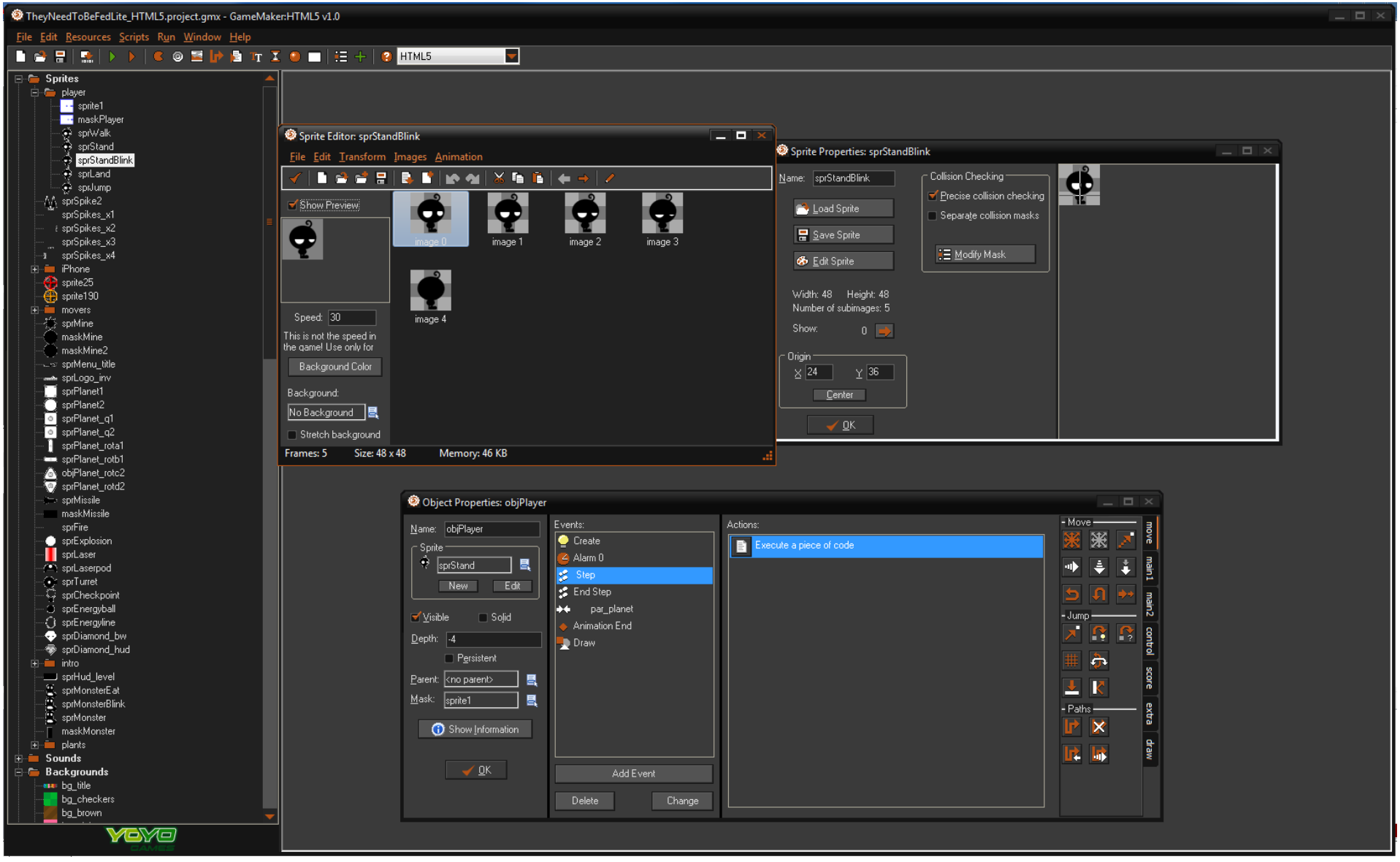
Loop 5 times times show complicated version

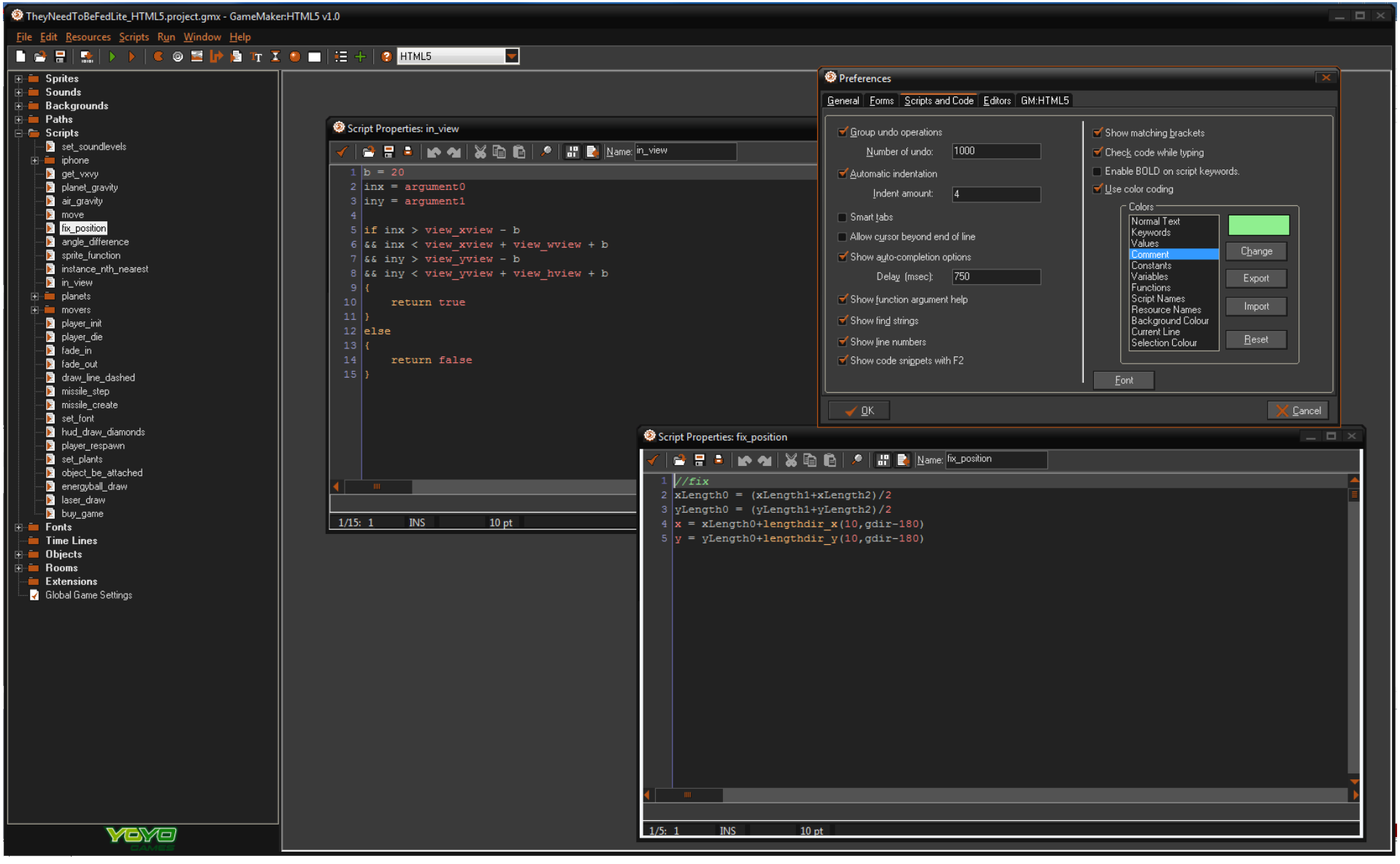
World set atmosphereColor to *duration* = BlinkDuration seconds *style* = abruptly more...
Light set color to *duration* = BlinkDuration seconds *style* = abruptly more...
World set atmosphereColor to *duration* = BlinkDuration seconds *style* = abruptly more...
Light set color to *duration* = BlinkDuration seconds *style* = abruptly more...

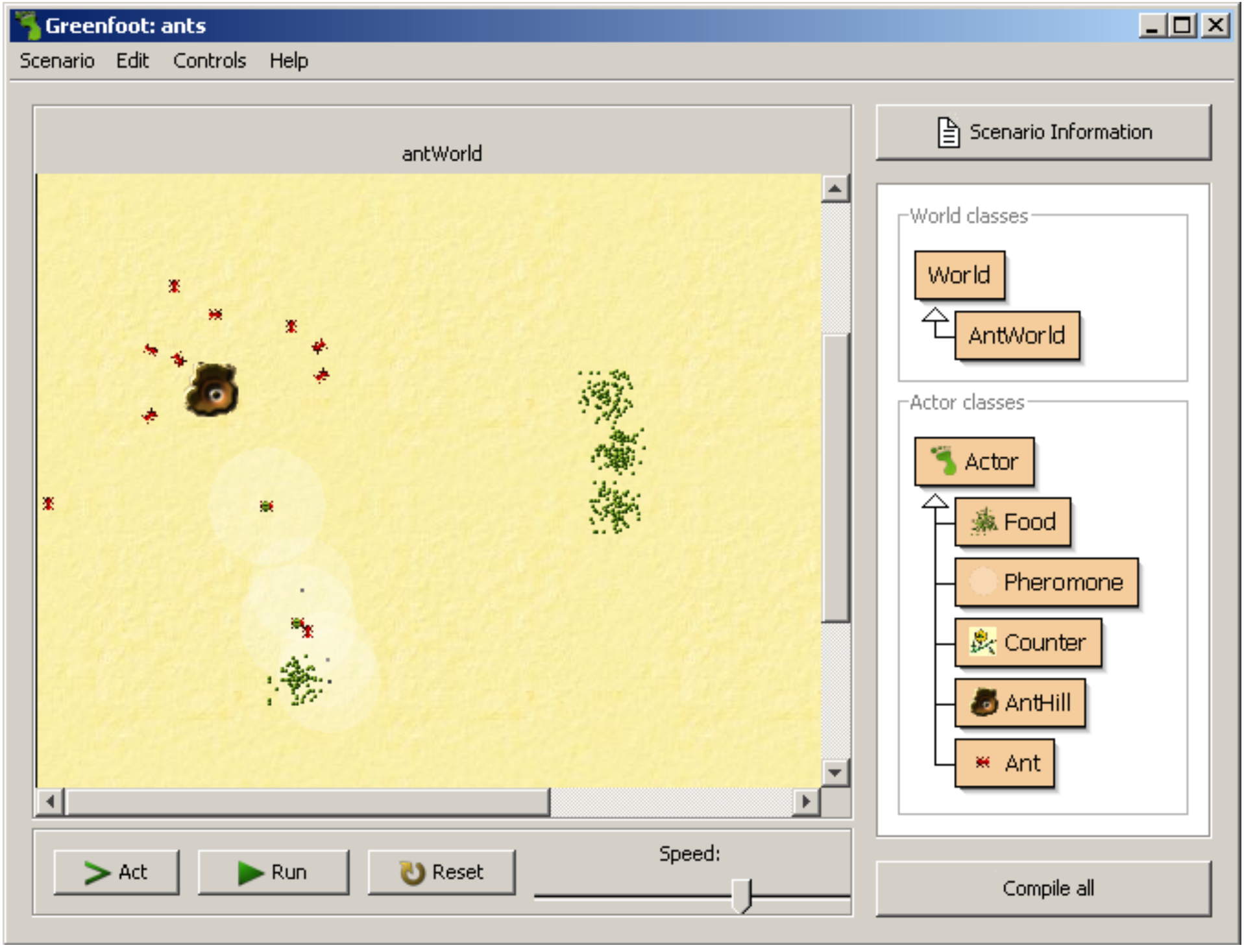
Biplane move to <None> *offset by* = Vector3(0, 0, 0) *duration* = 0 seconds more...
Biplane move up 3 meters *duration* = 0 seconds more...
Biplane stand up *duration* = 0 seconds more...
Camera move to <None> *offset by* = Vector3(0, 0, 0) *duration* = 0 seconds more...
Camera move backward 2 meters *duration* = 0 seconds more...
Camera move up 2 meters *duration* = 0 seconds more...
World set atmosphereColor to *duration* = 0 seconds more...
Light set color to *duration* = 0 seconds more...

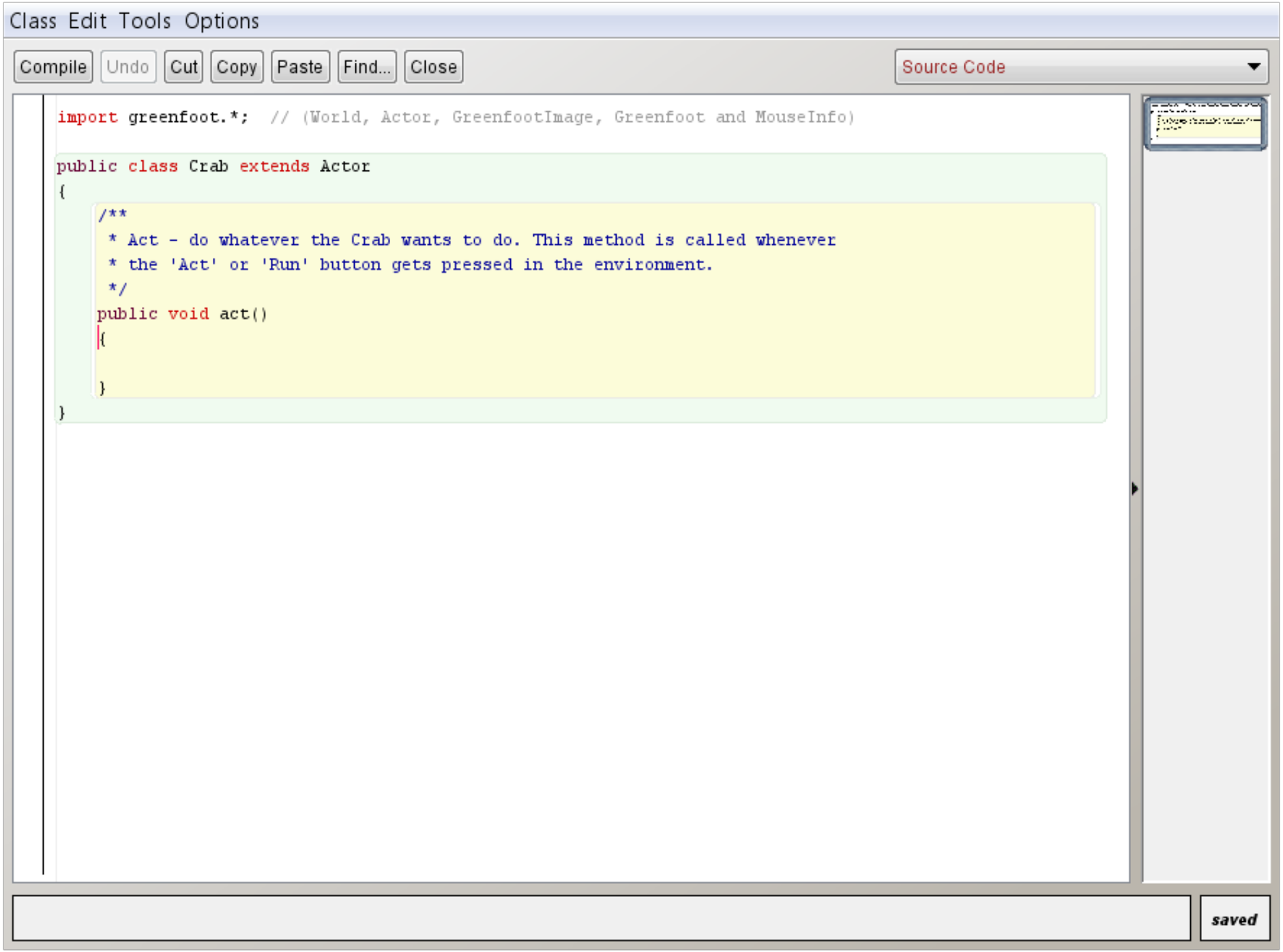
Else
Do Nothing


isothing.project.gmx* - GameMaker:HTML5 v1.0
File Edit Resources Scripts Run Window Help
HTML5
spr_player_ne
spr_player_nw
spr_player_sw_shadow
spr_player_se_shadow
spr_player_ne_shadow
spr_player_nw_shadow
spr_player_s
spr_player_n
spr_player_w
spr_player_e
Path Slabs
Grass Tiles
Graves
spr_shadow
sprite60
spr_grass_3d
spr_grass_shadow
sprite47
spr_path_3d_2
spr_church
spr_tree_0
sprite18
sprite80
NPCs
spr_gate
spr_fence_h
spr_fence_v
spr_fence_c
spr_blank
spr_weed
spr_arrow
spr_ghost_down
spr_ghost_up
spr_hole_1
spr_hole_2
spr_filled_1
spr_filled_2
spr_grassed_1
spr_grassed_2
Sounds
Backgrounds
Paths
Scripts
Fonts
Time Lines
Objects
Rooms
room0
room1
Room Properties: room1
Snap X: 32
Snap Y: 32
backgrounds
views
objects
settings
tiles
Object to add with left mouse:
obj_graveend
Left mouse button = add
+ <Alt> = no snap
+ <Shift> = multiple
+ <Ctrl> = move
+ <Space> = scroll room
Right mouse button = menu
+ <Shift> = delete all
+ <Ctrl> = delete
Middle mouse button = scroll
Delete underlying
x: 704
y: 160
object: obj_grass
id: inst_66572816
Firefox
Game Maker HTML 5
http://html5.yoyogames.com
Google
Yoyo Games
Bug Tracking
Mail
Shopping
Dailys
Bookmarks
HELP
RIP


TheyNeedToBeFedLite_HTML5.project.gmx - GameMaker:HTML5 v1.0
File Edit Resources Scripts Run Window Help
HTML5
Sprites
player
sprite1
maskPlayer
sprWalk
sprStand
sprStandBlink
sprLand
sprJump
sprSpike2
sprSpikes_x1
sprSpikes_x2
sprSpikes_x3
sprSpikes_x4
iPhone
sprite25
sprite190
movers
sprMine
maskMine
maskMine2
sprMenu_title
sprLogo_inv
sprPlanet1
sprPlanet2
sprPlanet_q1
sprPlanet_q2
sprPlanet_rota1
sprPlanet_rotb1
objPlanet_rotc2
sprPlanet_rotd2
sprMissile
maskMissile
sprFire
sprExplosion
sprLaser
sprLaserpod
sprTurret
sprCheckpoint
sprEnergyball
sprEnergyline
sprDiamond_bw
sprDiamond_hud
intro
sprHud_level
sprMonsterEat
sprMonsterBlink
sprMonster
maskMonster
plants
Sounds
Backgrounds
bg_title
bg_checkers
bg_brown
Sprite Editor: sprStandBlink
File Edit Transform Images Animation
Show Preview
image 0
image 1
image 2
image 3
image 4
Speed: 30
This is not the speed in the game! Use only for
Background Color
Background:
No Background
Stretch background
Frames: 5
Size: 48 x 48
Memory: 46 KB
Sprite Properties: sprStandBlink
Name: sprStandBlink
Load Sprite
Save Sprite
Edit Sprite
Collision Checking
Precise collision checking
Separate collision masks
Modify Mask
Width: 48 Height: 48
Number of subimages: 5
Show: 0
Origin
X 24 Y 36
Center
OK
Object Properties: objPlayer
Name: objPlayer
Sprite
sprStand
New Edit
Visible Solid
Depth: -4
Persistent
Parent: <no parent>
Mask: sprite1
Show Information
OK
Events:
Create
Alarm 0
Step
End Step
par_planet
Animation End
Draw
Add Event
Delete Change
Actions:
Execute a piece of code
Move
Jump
Paths
move main1 main2 control score extra draw


TheyNeedToBeFedLite_HTML5.project.gmx - GameMaker:HTML5 v1.0
File Edit Resources Scripts Run Window Help
HTML5
Sprites
Sounds
Backgrounds
Paths
Scripts
set_soundlevels
iphone
get_vxvy
planet_gravity
air_gravity
move
fix_position
angle_difference
sprite_function
instance_nth_nearest
in_view
planets
movers
player_init
player_die
fade_in
fade_out
draw_line_dashed
missile_step
missile_create
set_font
hud_draw_diamonds
player_respawn
set_plants
object_be_attached
energyball_draw
laser_draw
buy_game
Fonts
Time Lines
Objects
Rooms
Extensions
Global Game Settings
Script Properties: in_view
Name: in_view
b = 20
inx = argument0
iny = argument1
if inx > view_xview - b
&& inx < view_xview + view_wview + b
&& iny > view_yview - b
&& iny < view_yview + view_hview + b
{
return true
}
else
{
return false
}
1/15: 1 INS 10 pt
Preferences
General Forms Scripts and Code Editors GM:HTML5
Group undo operations
Number of undo: 1000
Automatic indentation
Indent amount: 4
Smart tabs
Allow cursor beyond end of line
Show auto-completion options
Delay (msec): 750
Show function argument help
Show find strings
Show line numbers
Show code snippets with F2
Show matching brackets
Check code while typing
Enable BOLD on script keywords.
Use color coding
Colors
Normal Text
Keywords
Values
Comment
Constants
Variables
Functions
Script Names
Resource Names
Background Colour
Current Line
Selection Colour
Change
Export
Import
Reset
Font
OK
Cancel
Script Properties: fix_position
Name: fix_position
//fix
xLength0 = (xLength1+xLength2)/2
yLength0 = (yLength1+yLength2)/2
x = xLength0+lengthdir_x(10,gdir-180)
y = yLength0+lengthdir_y(10,gdir-180)
1/5: 1 INS 10 pt


Greenfoot: ants
Scenario
Edit
Controls
Help
antWorld
Scenario Information
World classes
World
AntWorld
Actor classes
Actor
Food
Pheromone
Counter
AntHill
Ant
Act
Run
Reset
Speed:
Compile all


Class Edit Tools Options
Compile
Undo
Cut
Copy
Paste
Find...
Close
Source Code
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
public class Crab extends Actor
{
/**
* Act - do whatever the Crab wants to do. This method is called whenever
* the 'Act' or 'Run' button gets pressed in the environment.
*/
public void act()
{
}
}
saved