Chris Brett

Ocado Technology

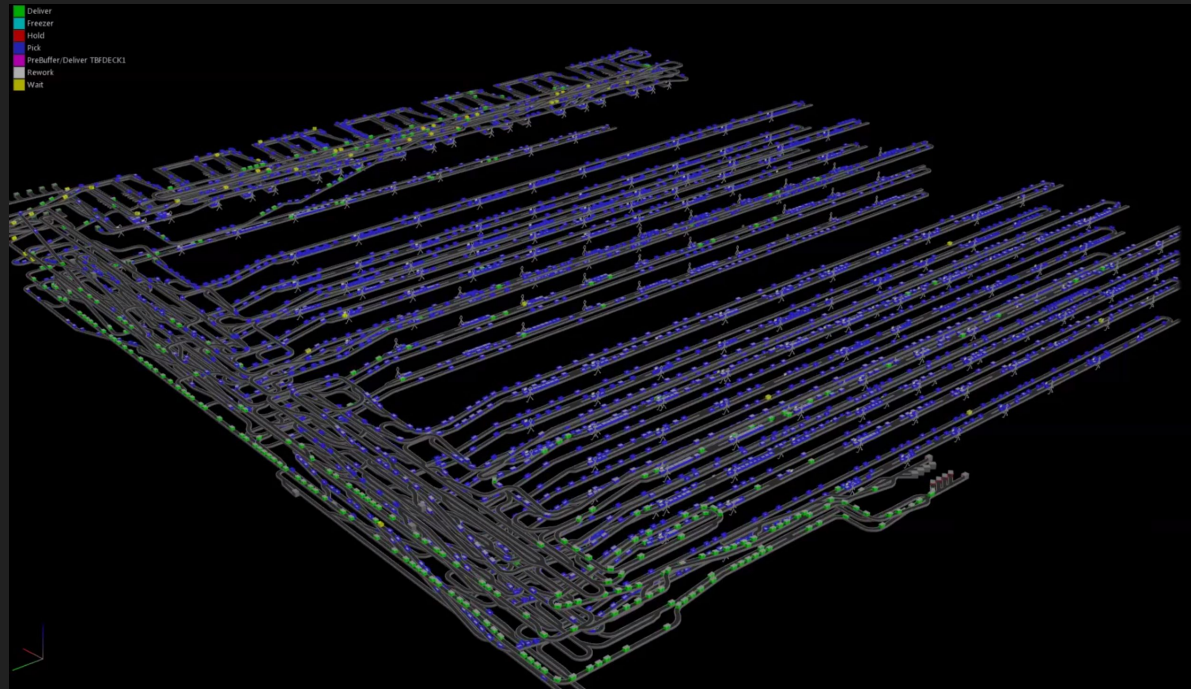
Creating gaming resources with an in-browser code interpreter and game engine

Day Job

Ocado Technology's Simulation Research Team Leader

Researching and developing warehouse automation solutions

Creating simulations to test against, to make predictions on KPIs, and to optimise against



CODE <FOR> LIFE

Aim: To use Ocado Technology's software engineering expertise to help teach kids to program.

The Code for Life initiative is...

Free – our way of giving back to the community

Volunteer driven – most contributors work in their own time

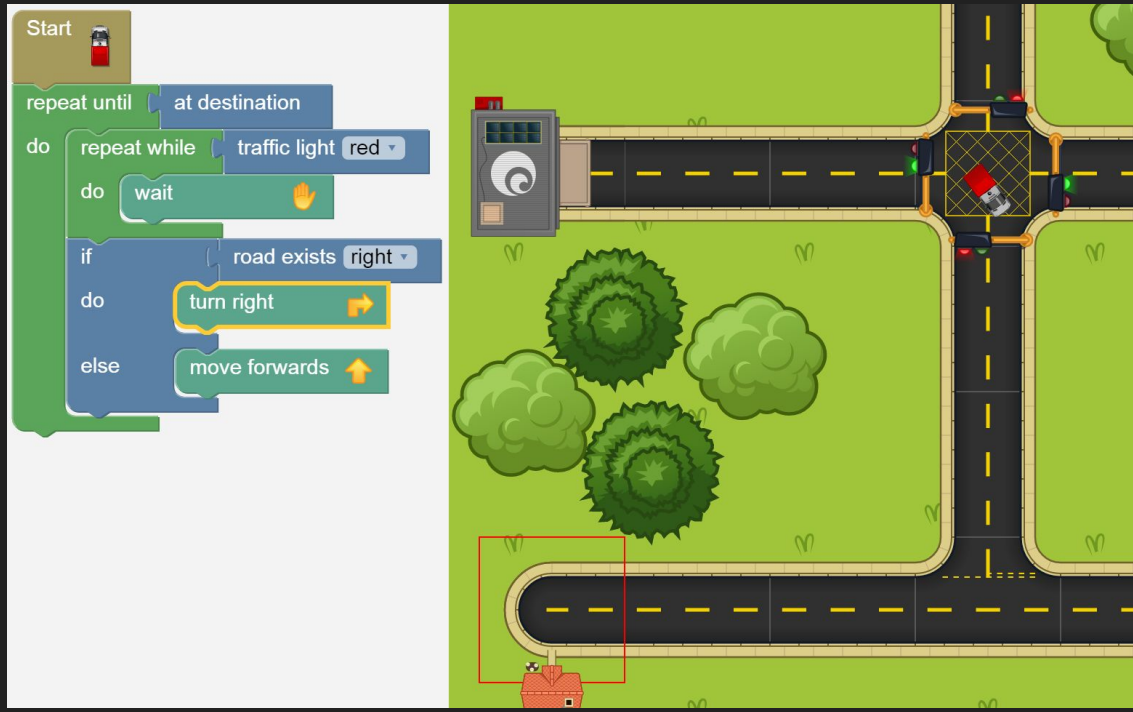
Open-source – get involved!

Rapid Router

First game

Teaches kids programming concepts

They use this to program a van to drive to its destination

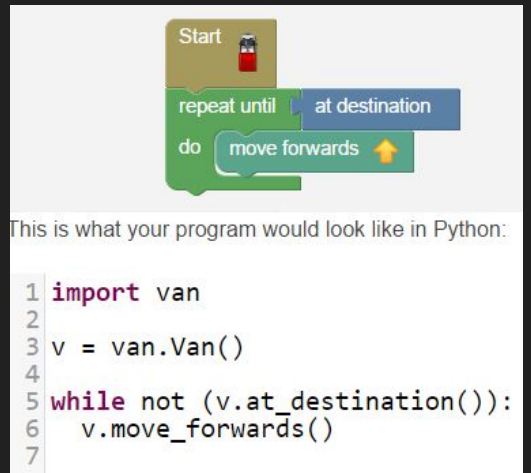


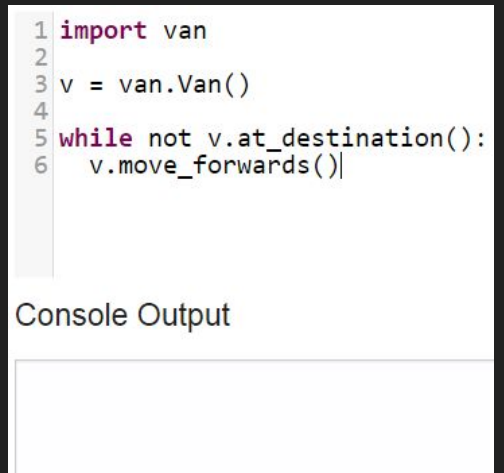
Rapid Router

Progressively introduces more complex programming concepts required to solve harder levels, or to solve similar levels in a more elegant way

Uses Blockly

Transitions kids from Blockly to Python





Demo

On to the tech...

User submitted code

- Input
- Execution

Game engine

Animation

User-submitted code - Where to execute

Server-side?

- Can the user affect the server it runs on?
  - Chroot jail
  - Separate VM or Container
  - PyPy sandbox
- Complexity
- Extra load

User-submitted code - Where to execute

Client-side?

- Isolated from server
- Potentially executable offline
- Must run in JS
- Security of user submitted code directly interacting with another user’s code (multiplayer)
- Results of the execution could be made up by the client - the server can’t trust them

User-submitted code - Blockly

Visual programming language written in JS by Google, similar to Scratch

Can attach Blocks together to form programs

Can inspect the player's code via JS:


```
var startBlock =  Blockly.mainWorkspace.getTopBlocks().filter(function (block) {
   return block.type === 'start';
})[0];

var nextBlock = startBlock.nextConnection;

var innerBlocksOfNextBlock = nextBlock.inputList;
```

Executing Blockly

Blockly now has better in-built execution support, but what we currently do…

Inspect blocks to compile into a “program” (list of commands, some of which have inner commands, e.g. if, while)

```
RapidRouter.BlocklyCompiler.prototype.createSequence = function(block) {
  var commands = [];
  while (block) {
    if (block.type === 'move_forwards') {
      stack.push(new ForwardCommanc(block));
    }else if (block.type === 'controls_repeat_while') {
      stack.push(this.createRepeatWhile(block));
    }...
    block = block.nextConnection ? block.nextConnection.targetBlock() : null;
  }
}

RapidRouter.BlocklyCompiler.prototype.createRepeatWhile = function(block) {
   var condition = this.getCondition(block.inputList[0].connection.targetBlock());
   var bodyBlock = block.inputList[1].connection.targetBlock();
   return new While(condition, this.createSequence(bodyBlock), block);
};
```

Executing Blockly

Run program to interact with game engine

```
while (thread.stack.length !== 0) {
  thread.stack.shift().execute(thread, model);
}
```

```
ForwardCommand.prototype.execute = function(thread, model) {
  queueHighlight(model, this.block);
  model.moveForwards();
};
```

```
While.prototype.execute = function(thread, model) {
  if (this.condition(model)) {
     thread.pushToStack([this]);
     thread.pushToStack(this.body.slice());
  }
};
```

User-submitted code - Python

Parse and execute Python in-browser, using Skulpt

Input via CodeMirror

- Syntax highlighting
- Code completion
- Can set code (initial / save + load):

```
codePanel.setValue(code);
```

```
1 import van
2
3 v = van.Van()
4
5 while not v.at_destination():
6   v.move_forwards()
```

Executing Python

Skulpt can parse and execute Python in JS:

```
Sk.importMainWithBody("<stdin>", false, codePanel.getValue());
```

Custom Python modules can be added to Skulpt

This is how it interacts with the game engine, e.g.

```
VanController.prototype.turn_right = function () {
    if (!Sk.failed) {
        this.queueHighlight();
    }
    Sk.failed = Sk.failed || !RapidRouter.model.turnRight();
};
```

Game Engine

OO JS - an application rather than script

```
RapidRouter.Model = function(nodeData, origin, destinations, maxFuel) {
    this.map = new RapidRouter.Map(nodeData, origin, destinations);
    this.van = new RapidRouter.Van(this.map.startingPosition(), maxFuel);
    ...
};

RapidRouter.Model.prototype.moveForwards = function() {
    var nextNode = this.map.getRoadForward(this.van.getPosition());
    return this.moveVan(nextNode, 'FORWARD');
};
```

The player's program is non-interactive - it can be run to completion instantly

The separation of engine and UI makes testing a lot simpler

Program Play-back

Executing the program adds animation events

```
RapidRouter.animation.appendAnimation({
    type: 'van',
    vanAction: action,
    fuel: this.van.getFuelPercentage()
});
```

After program execution, these are played back to the user

Animate the program stepping over code and the van moving in unison to aid understanding

Program Stepping

Highlights the section of the program “currently” being executed

Blockly:

```
block.addSelect();
```

Skulpt:

```
lineElement.style.background = colour;
```

Van animation

Considered Canvas, WebGL, settled on SVG (with Raphael)

Pre-created SVG images

```
paper.image(url, x, y, width, height);
```

Transformed on the fly

```
image.animate({transform: transformation}, duration, easing, performNextAction);
```

The animation is not very smooth on some mobile devices (a native mobile version is under development!)

This is probably due to the number of vertices in our SVG images…

Demo

New Game: AIMMO

In development

Code-named AIMMO: **A**rtificial **I**ntelligence **M**assively **M**ultiplayer **O**nline

Players code (in Python) their “avatar”, which competes with other avatars

Turn based

“King of the hill” style game mechanics

User code executed server-side in distributed, isolated, Kubernetes-managed docker containers

github.com/ocadotechnology/aimmo

CODE <FOR> LIFE

www.codeforlife.education

@chrisbrett665, @codeforlifeuk, @ocadotechnology

If you'd like to take a closer look at the code, or contribute, find us on GitHub:

github.com/ocadotechnology/rapid-router

github.com/ocadotechnology/aimmo

Questions?