Making Construals and the MCE

'Making construals' is the principal theme of the Erasmus+ CONSTRUIT! project. The acronym 'MCE' stands for the "**E**nvironment for **M**aking **C**onstruals".

The core language in the MCE is JS-Eden. This introduction complements a worksheet Getting Started with JS-Eden that was developed for use by schoolchildren.

CONSTRUALS AND REFERENTS

STEP 1: To invoke the MCE, click on the following url:

http://jseden.dcs.warwick.ac.uk/latest-master/index-dev.html?import=itag/solar/system/intro

Your display will have three windows, labelled **Script View**, **Canvas Picture** and **JSPE slides**.

Make sure that these are arranged so that you can see all three clearly on your screen.

- The Script View window is used for viewing and executing JS-Eden scripts.
- The Canvas Picture is used as an interface for visualisation and manipulation.
- The JSPE slides are used for presentation and documentation that can be closely integrated with JS-Eden scripts.

STEP 2: On the first JSPE slide, you will see a JS-Eden command:

```
import itag/solar/system;
```

and beneath it two underlined options 'submit' and 'copy to input'.

Press the **submit** option. This has the effect of executing the specified JS-Eden command. An animation will appear on the Canvas Picture.

`itag/solar/system` *is the name of a JS-Eden script. It is made up of definitions of observables. These*

Useful tips on using the MCE

Saving screen space: If you need to make more space on the screen, you can (e.g.) minimise the **JSPE Slides** window in the standard way. Minimised windows will either be represented by icons at the foot of the MCE display, or hidden, according to what option has been selected in the **Options** menu at the top of the MCE. Hidden windows can be recovered by clicking on their names in the drop down list associated with the adjacent **Existing Windows** option.

Editing and live editing: Observables can be redefined by editing their definitions, hovering in the gutter to the left of the observable name and clicking on the triangular 'enter' symbol that appears.

The Project Manager: Scripts can be accessed by interacting directly with the Project Manager. They can also be stored for private or public use *definitions are executed when we **import** the script.*

with a Google or Twitter login. For more details, consult this link.

STEP 3: To see some observables from the script, click on the **New Window** button at the top left of the MCE and select the '**Observable List**' option from the drop down menu. It will bring up a new window, labelled 'Observable List', with an empty box labelled *search* at the top. Type 'earth' into the search box: a list of observables appears.

The observables whose names are shown in green have values defined by dependency: hover over the mouse over such an observable, and you will see its definition (framed using the keyword '**is**'). The observable **earth** is a circle with centre at (**earthX**, **earthY**) for instance. Observables such as **earthDistance**, which are assigned an explicit value (using an '**=**'), are shown in black.

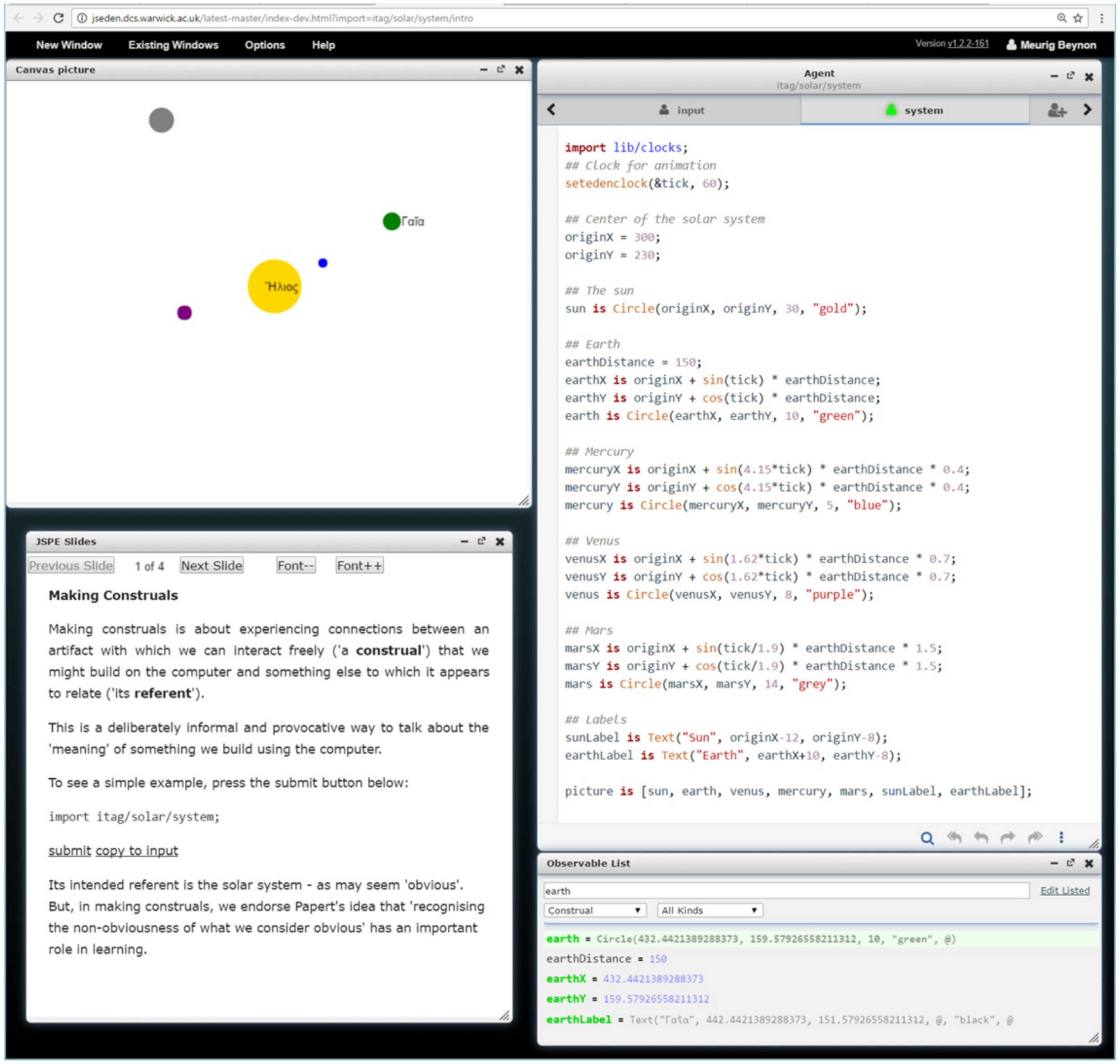
STEP 4: You can also inspect the whole of the itag/solar/system script:

The Script View has a tab inside it which is labelled **input**. Press the **copy to input** option on the first JSPE slide, ***then*** click on the label of the 'input' tab in the Script View. This copies the JS-Eden command

```
import itag/solar/system;
```

from the JSPE slide into the input tab

Select the **Inspect** menu option, represented by a spyglass icon on the left at the bottom of the Script View window. It will turn red. Click on the name of the itag/solar/system script, which is also displayed in red. A new tab, labelled 'system', appears in the Script View. The content of this tab is the full source of the `itag/solar/system` script.



Making Construals

Making construals is about experiencing connections between an artifact with which we can interact freely ('a **construal**') that we might build on the computer and something else to which it appears to relate ('its **referent**').

This is a deliberately informal and provocative way to talk about the 'meaning' of something we build using the computer.

To see a simple example, press the submit button below:

```
import itag/solar/system;
```

submit copy to input

Its intended referent is the solar system - as may seem 'obvious'. But, in making construals, we endorse Papert's idea that 'recognising the non-obviousness of what we consider obvious' has an important role in learning.

```
import lib/clocks;
## Clock for animation
setedenclock(&tick, 60);

## Center of the solar system
originX = 300;
originY = 230;

## The sun
sun is Circle(originX, originY, 30, "gold");

## Earth
earthDistance = 150;
earthX is originX + sin(tick) * earthDistance;
earthY is originY + cos(tick) * earthDistance;
earth is Circle(earthX, earthY, 10, "green");

## Mercury
mercuryX is originX + sin(4.15*tick) * earthDistance * 0.4;
mercuryY is originY + cos(4.15*tick) * earthDistance * 0.4;
mercury is Circle(mercuryX, mercuryY, 5, "blue");

## Venus
venusX is originX + sin(1.62*tick) * earthDistance * 0.7;
venusY is originY + cos(1.62*tick) * earthDistance * 0.7;
venus is Circle(venusX, venusY, 8, "purple");

## Mars
marsX is originX + sin(tick/1.9) * earthDistance * 1.5;
marsY is originY + cos(tick/1.9) * earthDistance * 1.5;
mars is Circle(marsX, marsY, 14, "grey");

## Labels
sunLabel is Text("Sun", originX-12, originY-8);
earthLabel is Text("Earth", earthX+10, earthY-8);

picture is [sun, earth, venus, mercury, mars, sunLabel, earthLabel];
```

```
earth
earth = Circle(432.4421389288373, 159.57926558211312, 10, "green", @)
earthDistance = 150
earthX = 432.4421389288373
earthY = 159.57926558211312
earthLabel = Text("Γαῖα", 442.4421389288373, 151.57926558211312, @, "black", @
```

*What you now have on the screen (taking into account the animation, the dynamically updating observables and the script of observable definitions) is 'a **construal**'. When we interact with it, we shall be thinking about what it is intended to refer to: "the solar system" as an independent external physical phenomenon ("its **referent**"). A construal exemplifies what Papert called 'an object-to-think-with'. We shall regard it as 'an **object-to-converse-with**': the JSPE slides provide an interface for talking about the construal and its referent.*

OBSERVABLES, DEPENDENCIES AND AGENCY

*Making construals is based on three fundamental concepts: **observables**, **dependency** and **agency**.*

These are the primary concepts that make the connections needed to support conversation with and about construals. Each person who views a referent makes their own construal in these terms.

ACTIVITY 1: Explore the observables in the solar system construal

1. inspect the definitions ot the observables **sun, earthDistance, mercury**, **mercuryX**, **originX, tick, picture** and **earthlabel**.in the `itag/solar/system` script.. For each observable, identify what they refer to, consider who the observer is, and in what context the observation is made,. What role might they play in conversations about the construal and the solar system?
2. Return to the JSPE Slides window and press the **Next Slide** button to move to Slide 2. Experiment by redefining the observables as explained on the slide. What can you learn about the significance of observables in this way?

ACTIVITY 2: Explore Dependency Map dependencies in the solar system construal

Click on the New Window button at the top left of the MCE and select the 'Dependency Map' option from the drop down menu. A **Dependency Map** window appears: enter the name 'mercury' into the search box at the top of the window, and you will see a directed graph whose edges show what the observable **mercury** (directly) depends on and what observables (directly) depend on it. To introduce an additional observable into the dependency map, tag a '|' symbol followed by its name onto the end of the search string (as in 'mercury|mercuryX' etc). Build up a complete picture of the dependency map for the observable 'mercury' in this way. Confirm that the dependencies are as you would have expected.

ACTIVITY 3: Explore agency in the solar system construal

An agent in the solar system is anything that can change its state. The most obvious example of an agent (and the only one that is internal to the construal) is the clock agent that advances the time. You can observe its effect by inspecting the value of '**tick'** in an Observable List. The clock agent is specified at the head of the `itag/solar/system script` . As has already been illustrated, much of the interesting agency in interacting with a construal is *what if?* style interaction carried out by human agents, all of whom are 'makers' of various kinds.

A group of actions that manipulate the clock agent is displayed on slide 3 of the JSPE presentation. Copy these to the input and experiment with them to see what each does. What happens if you simply press submit?

Slide 3 also includes a group of actions that can be used to manipulate the time directly. You can experiment with these in a similar way.

How would you use the above actions for manipulating the clock and time to decide how closely one tick in the construal approximates a day in the referent? How might you be able to verify directly that Earth and Mars are nearest to each other every two years or so?

REFLECTIONS

Slide 4 of the JSPE presentation shows how the simple construal can be adapted to account for the observed ''recession of the planets. Though it may initially seem 'obvious' that the construal 'represents' the solar system, in fact the main role of making a construal is to expose the limitations in any representation of a referent. There are inumerable features of the planetary system that are not reflected in the construal: elliptic orbits, moons, forces, magnetic fileds, three dimensional dispositions and rotations that accountfor days and seasons etc. The virtue of making construals is that it is possible to embellish the observational model so as to embrace other perspectives, The Mars construals, developed by Rusell Boyatt, represent a step in this direction, supporting richer modes of visualisation and making use of actual data about locations. These construals feature in the Project List which can be aceessed from the drop down menu associated with the **New Window** button.