App Inventor (Blockly-based)

Andrea Sterbini – sterbini@di.uniroma1.it

App Inventor: an IDE to design and program an Adroid app

Built with Blockly http://ai2.appinventor.mit.edu

Build, compile, and deploy on the phone an Android App

Automatic deploy changes either to the Phone or to an Emulator

- Install AI2 Companion App

Packaged apps can be installed stand-alone on the phone

Note: to use it in Genymotion install the Arm Translation package

App structure

One "screen" for each phase (config, login, play levels, results ...)

Screens are independent and DO NOT share data or code

(but a local DB component allows to exchange data)

You can pass/retrieve some text when switching to another screen

Apps are independent and DO NOT share data or code

(here you can exchange data by using an external WebService/WebDB)

Resources (video, audio, data, files, images etc) are bundled

Suggested Limit: <u>10 screens</u> max

To mimic many more screens you can hide/show parts of the App

Many widgets/objects available

Fields:	Form fields and automatic layout constraints
Media:	Sound, Movie, Camera, SoundRecorder, SpeechRecognizer, TextToSpeech, …
Drawing:	Canvas, Sprite, Ball
Maps:	Maps, polygonals, Markers, Features (GIS)
Sensors:	Accel, Temp, Gyro, Barcode, Pedometer, NFC, …
Social:	Contacts, PhoneCall, Email, Twitter, Sharing, Texting
Storage:	TinyDB, TinyWebDB, FusionTables, File
Connect:	BT Client, BT Server, Web, Activity
Lego:	NXT, EV3

Execution model: event-based programming

NO concurrent events can be defined (no parallelism)

NO message passing

Many objects generate events

- E.g. “When the screen changes”, “When the button is clicked”, “When the text-area content is changed”

Async protocols are split in 2 or more phases

- E.g. “Ajax query to web URL” ==> “When the response arrives” event
- This to remove busy wait and to get an async interaction
- To behave differently in different cases you can use globals as semaphores

NO object orientation (no way to add properties or to clone)

Data types

Numbers, Strings, Lists, Lists of Lists, (Booleans)

All interface widgets are objects with:

- Predefined Properties (pre-set in the IDE, or read/changed by program)
- Events they can generate
- Methods that can be called

Some objects are not visual (i.e. BluetoothClient, Sound, …)

Computed values are represented with a “puzzle” connector (while in Scratch they were ovals)

No data types are enforced

Code style

You implement mainly Events, Procedures and Functions

- Functions are “special”, they return a value

GLOBAL variables outside any Event/Function/Procedure

You can define LOCAL variables

- They can be changed/used only within their “scope bracket” (or can used as a return value)

This allows some kind of “functional programming” style

You can “collapse” the functions/events/procedures

You can enable/disable some blocks

Nice trick to enable cooperation

Ask each student in a group to implement just one screen of a complex App

Initially you prepare and distribute a template App with the desired empty screens

At the end you merge into a single App the screens made by the students (with the AI2 Merger app)

Common resources can be shared among screens