eSeeCode: exercitant la lògica a través de la programació

Joan Alemany Flos[1], Jacobo Vilella Vilahur[2], Laura Morera Úbeda[3] , Marc Guinjoan Francisco[4]

[1] eSeeCode, eXplorium Serveis educatius, jalemany@eseecode.com
[2] eSeeCode, jvilella@eseecode.com
[3] eXplorium Serveis Educatius, Universitat Autònoma de Barcelona, laura@explorium.cat
[4] eXplorium Serveis Educatius, Universitat Autònoma de Barcelona, marc@explorium.cat

Resum de la comunicació

La programació és una disciplina que ens permet treballar molts aspectes fonamentals de les matemàtiques. A part del context propi del problema que es vol resoldre (de geometria, de teoria de nombres, etc), s'usa constantment la lògica, l'estructuració del pensament i el rigor.
eSeeCode (www.eseecode.com) és un llenguatge de programació de lliure accés pensat per ser portat a classe en diferents etapes escolars per treballar aquests conceptes. El seu disseny permet afrontar la programació des de diferents punts de vista, des d'una programació totalment visual, fins a escriure codi concret en llenguatge de programació formal.
En aquesta presentació parlarem dels aspectes que tenen en comú la programació i les matemàtiques i per què és necessari que els professors de matemàtiques portin aquesta disciplina a classe. També mostrarem casos reals d'alumnes i experiències didàctiques realitzades amb eXplorium.

PARAULES CLAU: innovació pedagògica, resolució de problemes, programació.

Per què programació a la classe de matemàtiques?

Han passat gairebé 50 anys des de l'aparició dels primers llenguatges de programació educatius. L'any 1967 es va crear el llenguatge Logo, un llenguatge de programació enfocat a l'entorn escolar. El que va ser un moment revolucionari va acabar a l'oblit amb el temps.
Als anys 80 es van realitzar proves pilot en diverses escoles de Catalunya, i posteriorment es va implementar en d'altres centres. Tanmateix cap a l'any 2000 l'ús del Logo a classe va anar minvant. Actualment s'ha recuperat aquest interès a través de l'aprenentatge i l'ús de l'Scratch, però s'ha vinculat més a la branca de tecnologia que a la de matemàtiques.

Hi ha molts aspectes positius a destacar de l'experiència de treballar amb Logo. Per una banda, la construcció i creació per part de l'alumne, però també el pensament crític, lògic i rigorós. Aquests són aspectes fonamentals de l'ensenyament de les matemàtiques.

Actualment Scratch és l'eina de desenvolupament preferida a les escoles. Aquest és un llenguatge visual creat per Resnik l'any 2005, i que intentava modernitzar els conceptes que es treballaven amb Logo. Tanmateix la implementació a Espanya ha vingut donada a les classes sobretot de tecnologia, i s'han perdut alguns dels aspectes claus de l'enfocament que tenia el Logo. El fet de ser un llenguatge de blocs facilita a l'alumne començar a programar traient així la por del "full blanc" que produeixen els llenguatges textuals. Aquest canvi d'entorn ha potenciat la part creativa de la programació, es poden fer exercicis més visuals, animacions, i videojocs de manera més senzilla, però ha perdut en la part lògica i la rigorositat.

La metodologia amb la que es treballa el Scratch també ha canviat. Els alumnes no treballen tant l'aspecte seqüencial com l'orientació a objectes. Això fa que perdin la perspectiva del flux del programa. També l'actitud de l'alumne davant un programa que no fa el què es vol acaba sent una aproximació per prova i error en comptes d'un anàlisi crític. Això és degut a la facilitat de canviar els programes i executar-los. Tot i que aquests aspectes poden ser interessants a treballar en una assignatura com tecnologia, es pot caure en la temptació de deixar perdre l'oportunitat de treballar l'estudi de l'algoritme a través de l'anàlisi del codi, donant tota la importància al resultat i molt poca al procés.
Per altra banda la rigorusitat també es redueix, ja que no és possible fer errors sintàctics.

Els conceptes que es treballen quan resolem problemes de matemàtiques poden reforçar-se amb la programació. La metodologia que proposem està inspirada en la clàssica de Polya de resolució de problemes:

1. Plantejament del problema.
2. Anàlisi del problema "sobre paper". Realitzar els càlculs necessaris, dibuixar els esquemes, escriure les coordenades dels punts més importants, etc.
3. Disseny de l'algorisme

4. Implementació a l'ordinador. Per escriure el codi utilitzem la metodologia *Baby steps*
5. Comprovació del resultat (a vegades visual)
6. Valoració i retorn als punts 1, 2 i 4.

L'avantatge d'utilitzar la programació és que es treballa el rigor (si no és correcte no s'executa), i la comprovació del resultat és més directa que quan treballem problemes purament matemàtics.

Presentació de la plataforma eSeeCode.

eSeeCode és al mateix temps una plataforma i un llenguatge de programació que recupera els aspectes positius del Logo i el modernitza. És un llenguatge educacional basat en JavaScript, i que es pot estendre fàcilment usant aquest.

L'entorn d'eSeeCode ofereix 4 nivells de programació diferents, des d'una interfície purament gràfica a un editor textual. Aquests nivells els anomenem vistes perquè són representacions d'un mateix codi (figura 1) i l'alumne pot passar en tot moment d'una vista a una altra. D'aquesta manera l'entorn de programació s'adapta al nivell d'aprenentatge de l'alumne i pot permetre al professor centrar-se en els problemes i no tant en el llenguatge.

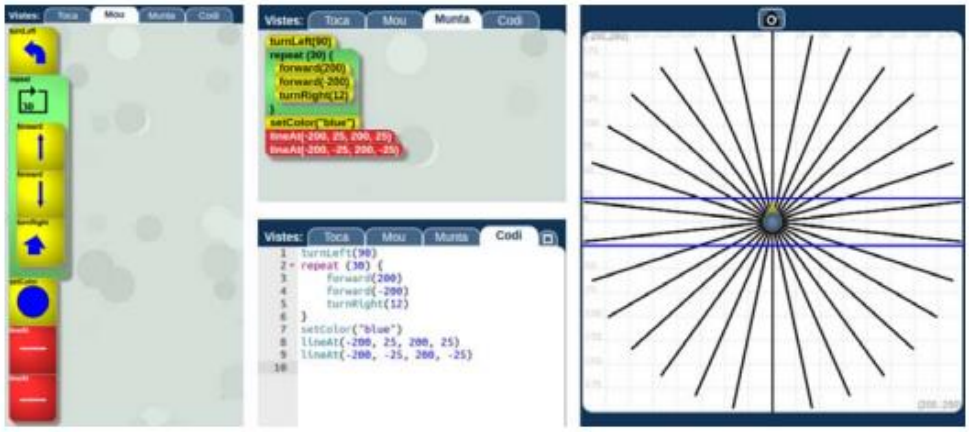


Figura 1: Diferents vistes d'un mateix codi, i el resultat.

La vista *Toca* és la nostra aproximació a la resolució de puzles i està dissenyada per alumnes de 5 a 8 anys. En aquesta vista el nombre d'instruccions disponibles és més reduït i estan representades només per icones (figura 2) que s'executen quan l'estudiant les prem. Per tal de facilitar l'ús amb alumnes del primer cicle de primària les icones no tenen noms escrits i cobreixen només els moviments de la "guia", la mida de les línies, els colors i el posicionament general. D'aquesta manera es poden treballar problemes de seqüencialitat.

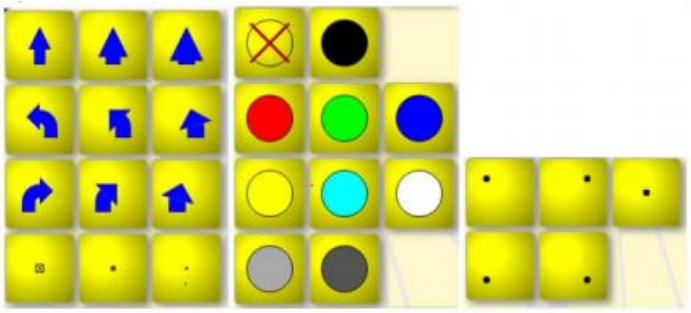


Figura 2: Instruccions de la vista *Toca*.

La segona vista és la vista *Mou*. En aquesta vista els alumnes poden moure lliurement les instruccions, que accepten diferents arguments. Les icones s'adapten als valors dels arguments. Un exemple d'aquest comportament el tenim a la Figura 3. En aquesta etapa les instruccions inclouen noms ja que així l'alumne pot començar a identificar les instruccions amb paraules i acostumar-se al llenguatge. Per executar el programa l'alumne ha de prémer el botó d'execució.

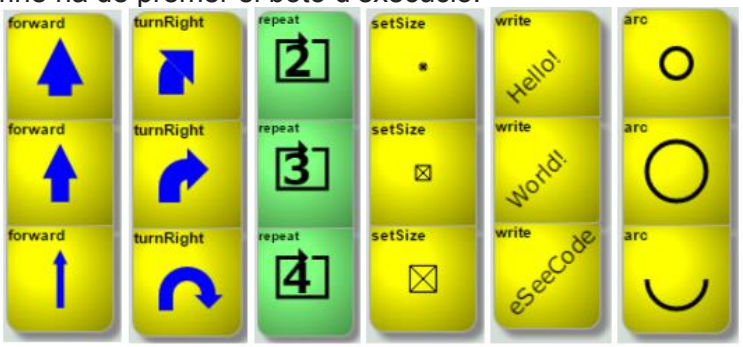


Figura 3: Les icones de la vista Mou s'adapten depenent del valor dels seus arguments

La següent vista és la vista *Munta*, que és similar a la vista *Mou*, ja que els blocs es poden desplaçar lliurement per l'àrea de codificar, però els blocs ja no són icones sinó que estan formats pel nom de la instrucció i el valor dels seus paràmetres. L'estudiant pot llegir el codi de manera completa, però per programar ha d'escollir les instrucions entre una llista de possibilitats. D'aquesta manera s'evita inicialment l'efecte de la pàgina buida que bloqueja els alumnes quan programen.

L'última vista és la vista *Codi*. En aquesta els alumnes poden escriure el seu propi codi. Com que eSeeCode està creat utilitzant JavaScript, en aquesta vista també es poden utilitzar directament primitives d'aquest llenguatge. D'aquesta manera l'alumne pot arribar a aprendre un llenguatge de nivell professional. En aquest punt, el llenguatge permet errors sintàctics i t'avisa quan intentes executar un programa. D'aquesta manera tornem a treballar el punt de rigor en l'ús d'un llenguatge informàtic. Per tal de facilitar la feina a l'alumne i ajudar-lo a millorar en la presentació del codi, el propi entorn neteja i fa un reestil del codi.

En el context de "terra baix, sostre alt" que proposa Papert i que Resnick utilitza, eSeeCode té un sostre més baix que Logo i Scratch, i un sostre més alt, podent-se comparar a llenguatges professionals com són C++ i Python.

Per tal d'enfocar eSeeCode a la resolució de problemes s'ha incorporat una manera fàcil de llegir i escriure valors per pantalla, podent treballar fàcilment tant amb

problemes geomètrics com amb problemes numèrics. L'entorn també disposa d'un sistema avançat per tal de depurar el codi.

Com triar un bon problema?

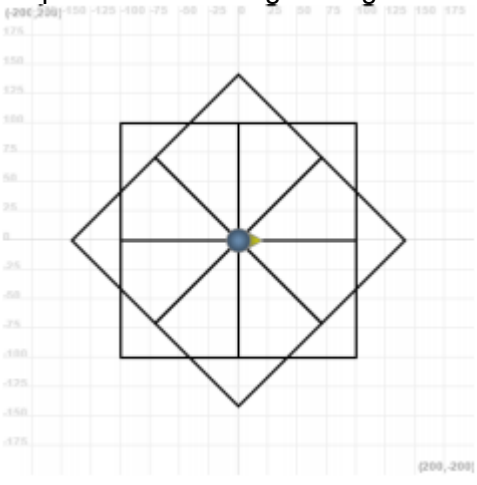
En el moment de preparar problemes de l'àmbit de l'algorísmica i de les matemàtiques és important tenir en compte diversos aspectes, des de l'elecció del context fins al propi contingut de les preguntes, és a dir, els objectius didàctics del problema. Hem de tenir en compte que la majoria de problemes accepten múltiples algoritmes com a solució, però que alguns seran difícils a l'hora de programar. És per això que és important tenir clar quines dificultats es trobarà l'alumne a l'hora de triar un algoritme en front a un altre, per tal de poder guiar-lo adequadament quan es trobi amb dificultats.

La clau al bon ensenyament és plantejar exercicis que guiïn a l'alumne per fer-li arribar als objectius desitjats preveient les dificultats que s'hi trobarà. Per tal d'il·lustrar-ho proposem tres exercicis pensats per treballar amb alumnes amb diferents nivells de coneixement informàtic.

Per començar en els cursos de primària i al primer cicle de secundaria és essencial que el context sigui visual amb l'objectiu de facilitar la comprensió (intuïtiva) de les instruccions i l'anàlisi del codi.

Enlloc de presentar un problema, parlarem d'objectius a assolir. En aquest cas el que ens interessarà serà l'arbre per arribar a aquest objectiu (Figura 4) i com adaptar cada objectiu parcial a les necessitats dels alumnes.

Objectiu 1: Fes un programa per dibuixar la següent figura:



Abans d'enfrontar-se amb aquest repte, a l'alumne li demanarem de fer uns objectius parcials, per tal de veure que va entenent el procés (i s'acostuma a l'entorn).

Entre d'altres preguntes demanaríem que ens dibuixes un quadrat, que optimitzés el seu codi afegint la instrucció *repeat*, etc. En aquest punt és important fer entendre a l'alumne els conceptes d'algoritme i de resultat, i que un mateix "problema" pot tenir

diferents solucions. Per exemple els dos programes que dibuixen quadrats són equivalents.

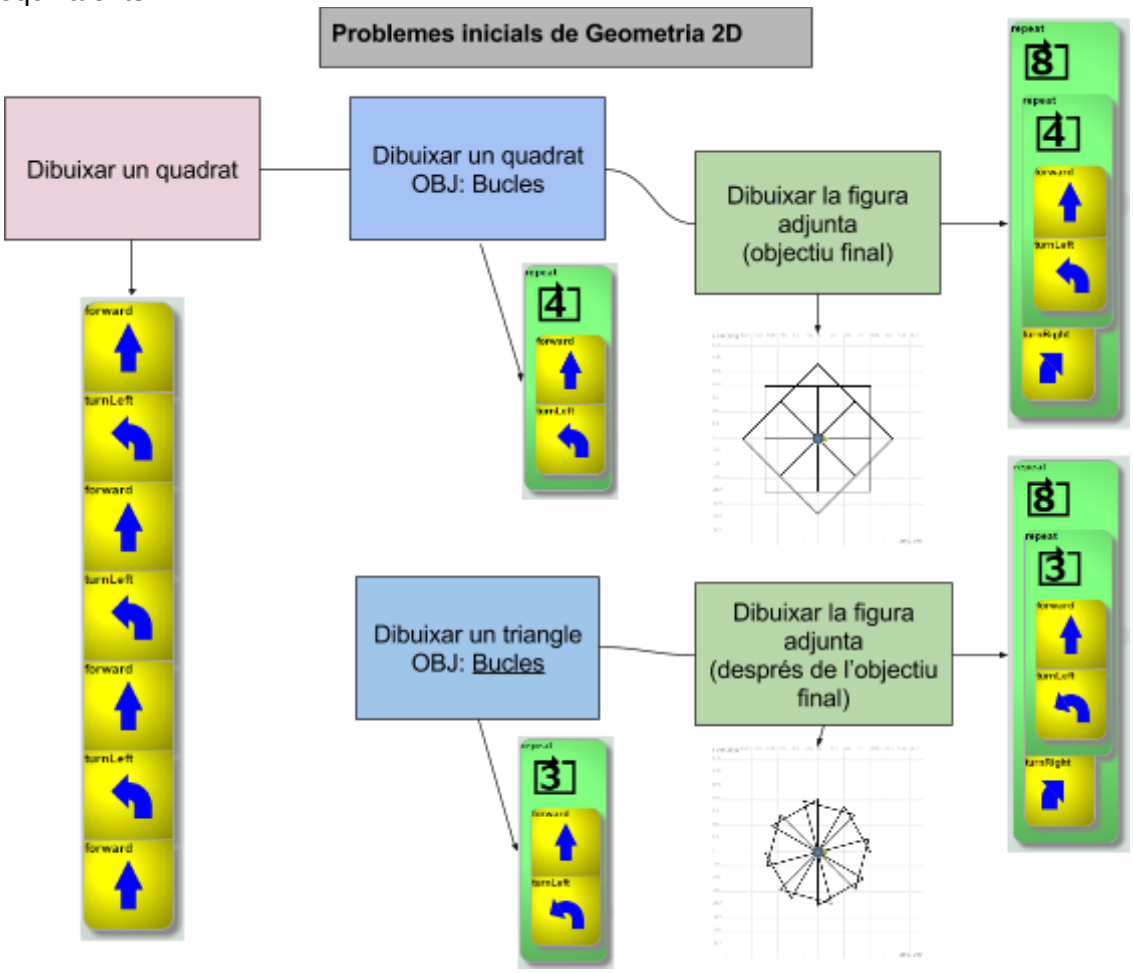


Figura 4: Esquema de plantejament i resolució d'un problema bàsic de geometria 2D (Vista de Mou).

La primera pregunta proposa un problema fàcil d'entendre i fàcil d'executar per ajudar a l'usuari a què s'acostumi amb l'entorn. D'aquesta manera dibuixar un quadrat té com a únic objectiu aprendre el funcionament de eSeeCode.

Tot seguit s'introdueix la noció de les repeticions (bucles). Per això s'explica la instrucció *repeat* i es demana resoldre el mateix exercici.

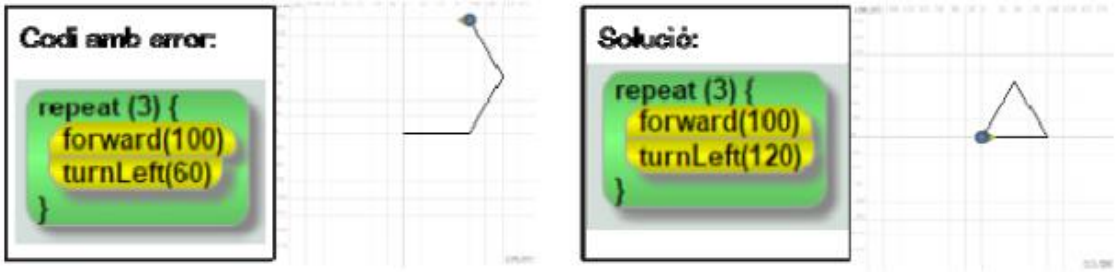


Figura 5: Els triangles no es dibuixen amb girs de 60 graus (gir interior) sino de 120 graus (gir exterior).

Quan intentem fer el mateix problema però amb el triangle trobem una dificultat afegida que és l'angle de gir. En aquest cas molts dels participants giren 60 graus erròniament, quan n'haurien de girar 120 graus (Figura 5): Tanmateix, si és un problema que es fa amb alumnes de primària, és un bon moment per introduir el concepte d'angle suplementari. Així creem un espai natural on aquest angle és útil, i no l'introduïm de manera purament teòrica.

Per la segona part del repte podem veure dues opcions amb enunciats molt similars. Per un costat proposem fer la figura final (quadrats) i si fa falta la de triangles (Figura 6). En aquest repte no es donarien els patrons marcats.

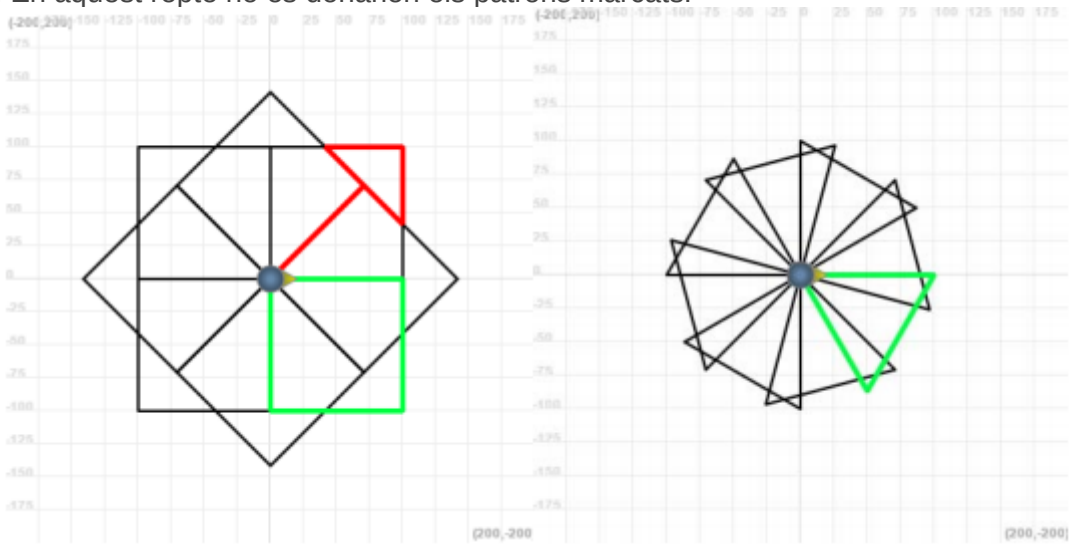


Figura 6: El resultat de realitzar la mateixa operació amb quadrats i amb triangles i possibles patrons.

Tot i que semblen problemes que treballen els mateixos conceptes, no és així. Mentre que en el quadrat l'objectiu és treballar el reconeixement de patrons, en el segon cas l'objectiu és treballar bucles. Això és degut que en el quadrat els alumnes troben dificultats en identificar els patrons que componen la figura. Tenen diverses possibilitats, i se'ls fa difícil identificar les més simples de programar. En canvi en el segon cas els alumnes detecten ràpidament quin és el patró (triangle) que es repeteix i com es repeteix, i ho programen de manera més ràpida i eficient. Si donem la segona figura abans que la primera ens trobarem que l'alumne troba el patró del quadrat de manera més ràpida, i per tant perdríem l'opció de treballar el reconeixement de patrons, per tant ens podríem guardar aquesta figura per donar-ho com a ajuda.

Quan treballem amb alumnes més grans ja podem parlar de problemes concrets. Un exemple de problema no visual que es pot treballar amb alumnes de 2n cicle de ESO / Batxillerat és el següent:

Problema 2: Escriu els nombres primers de 1 a 100. Es diu que un nombre és **primer** quan només té 2 *divisors*.

Primerament cal veure que en el propi problema es donen les definicions necessàries per tal que aquest pugui ser autoexplicatiu. Per tal d'ajudar a estructurar el treball i

arribar a la solució de manera natural, proposem treballar amb els següents subproblemes (Figura 7):

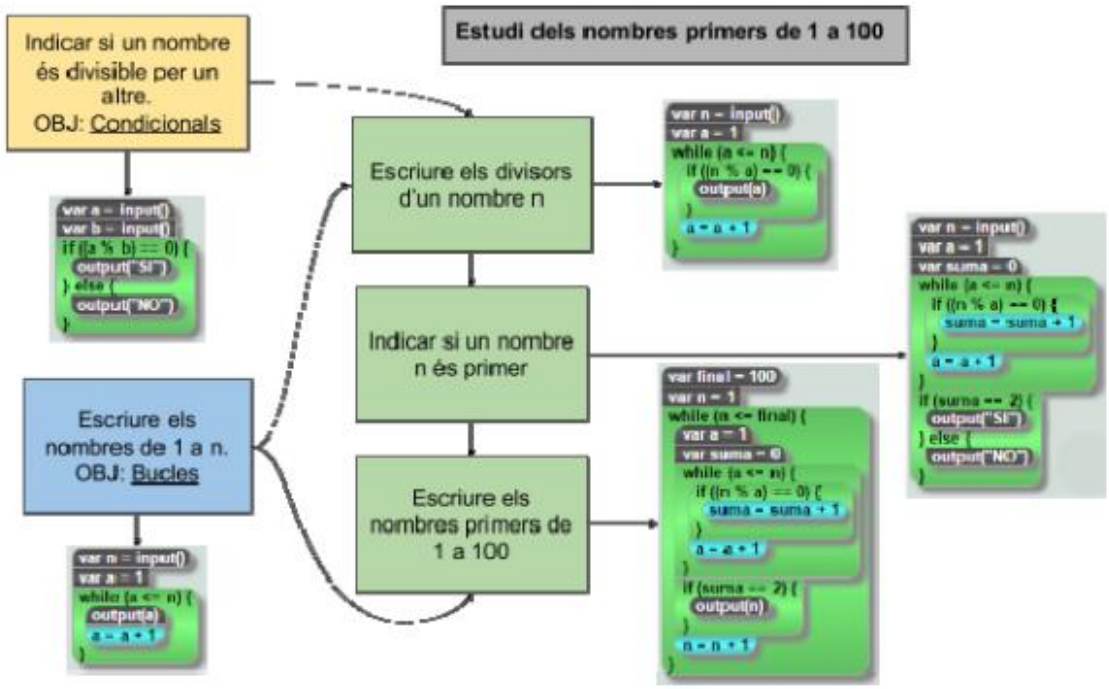


Figura 7: Procés de resolució del Problema 2 com a composició de subproblemes i les seves solucions.

Observant a la Figura 7 el codi de cada subproblema es visualitzen les similituds de les solucions i l'evolució d'una solució a una altra.

Quant al plantejament del problema a classe, similar a l'exemple geomètric, l'alumne ha de poder superar tot sol els primers dos subproblemes. En aquest cas estem suposant que l'alumne té uns coneixements mínims del llenguatge. Alternativament, aquests dos primers subproblemes poden ser posats d'exemple per tal que l'alumne construeixi el tercer: Escriure els divisors d'un nombre n. Com es pot veure i per facilitar l'expressió algorísmica, la propia definició de nombres primers que es dóna té una forta correlació amb l'algoritme esperat pel quart dels subproblemes.
Finalment si apliquem dos cops els conceptes, podem arribar a la solució final del problema original.

Un cop l'alumne té un domini tant de l'eina com del llenguatge, es poden proposar problemes més globals on els alumnes puguin fer un ús tant de la part numèrica com de la part gràfica. Un exemple seria el següent:

Problema 3: Fes un programa que trobi el màxim comú divisor entre dos nombres seguint l'algoritme d'Euclides.
Genera una interpretació visual de l'execució i estudia la complexitat del teu programa.

Tot i que hem suposat una experiència prèvia amb l'eina, hi ha una primera part de la solució d'aquest problema que seria la traducció literal de l'algoritme a codi màquina. El valor afegit que poden donar les matemàtiques en un problema com aquest seria

treballar la demostració de que l'algoritme funciona. En particular es pot intentar demostrar que l'algoritme acaba en un nombre finit de passos (Figura 8).

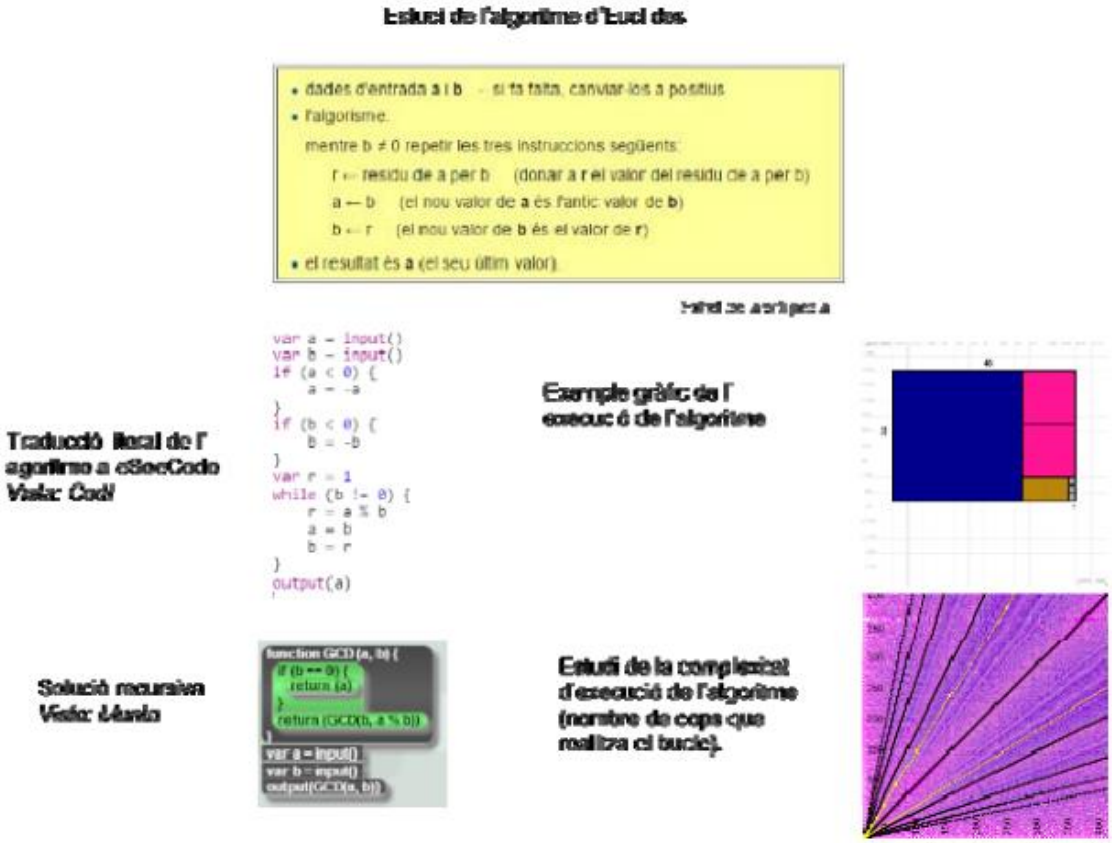


Figura 8: Estudi de l'algoritme d'Euclides i de la seva implementació

La facilitat amb la que es pot treballar visualment amb eSeeCode permet convertir un problema numèric com aquest en un treball de recerca per part de l'alumne. Per exemple, en l'estudi de l'execució de l'algoritme es pot detectar que hi ha unes parelles de nombres que sistemàticament triguen més que les seves veïnes (en groc a la 4a imatge de la Figura 8). Explorant una mica més es pot descobrir que aquestes parelles on l'algoritme fa més passos segueixen la proporció àuria, ja que són nombres consecutius de la successió de Fibonacci.

Casos d'alumnes i experiències a eXplorium

eSeeCode s'ha utilitzat amb èxit en proves pilot a diferents centres de Barcelona i com a complement d'activitats extraescolars amb eXplorium Serveis Educatius.

D'una banda s'ha utilitzat com a eina per la transició d'Scratch a C++ aprofitant la seva flexibilitat a l'hora de visualitzar el codi. Els problemes proposats als alumnes han estat sobretot per treballar la geometria 2D ja que posteriorment amb C++ aquesta ajuda visual no és senzilla de treballar. La resposta dels alumnes ha estat molt positiva, superant les expectatives dels professors. A diferència de quan no s'utilitzava aquesta plataforma, l'alumne ha sabut passar a treballar amb codi textual de manera natural, ja que ha vist els avantatges de poder escriure el seu el propi codi.

D'altra banda s'ha utilitzat per ensenyar a programar a alumnes que no tenien coneixements previs de programació, en particular s'ha inclòs eSeeCode en diversos tallers d'introducció a la robòtica per tal d'ensenyar els conceptes bàsics de la programació. La metodologia seguida en aquest cas ha estat demanar als alumnes dibuixar el quadrat i el triangle com en el primer problema proposat. Després se'ls hi ha ensenyat el concepte de repetició. D'aquesta manera els alumnes han tingut una explicació dels entorns de programació i s'ha pogut passar a programar els robots amb l'entorn EV3 de Lego. En aquest punt s'ha pogut fer èmfasi en el pas del món virtual (on tot sembla exacte) al món real.

En la presentació d'aquesta comunicació al congrés es mostraran casos d'alumnes resolent aquests problemes que hem detallat i farem un anàlisi dels errors més freqüents i les diferents estartègies que utilitzen els alumnes per tal de resoldre el mateix problema.

Conclusió: Reptes actuals a l'ensenyament de la programació

L'ensenyament de la programació és una disciplina que no pot quedar únicament sota la responsabilitat de l'assignatura de tecnologia. Els professors de matemàtiques tenim una oportunitat i una responsabilitat de treballar conceptes algorísmics a classe, fent una bona tria de quins problemes i reptes proposem.

Oportunitat, perquè podem treballar de manera amena la resolució de problemes complexes, ja sigui per tractar-se de nombres grans, o de problemes amb una forta calculabilitat. També s'ha de valorar l'aspecte motivant que genera la programació, avui en dia, en l'alumnat. Responsabilitat, ja que l'ensenyament d'aquesta disciplina requereix del pensament matemàtic: un pensament rigorós que fuig de l'aproximació per prova i error, i que va més enllà intentant demostrar els seus continguts, i plantejar noves preguntes.

Treball emmarcat dins del projecte atorgat per la Generalitat de Catalunya a l'equip GIPEAM - Grup d'Investigació en Pràctica Educativa i Activitat Matemàtica, amb referència SGR2014-972

Referències

Code.org (2013). http://www.code.org

eSeeCode (2015-2016). http://www.eseecode.com

Lewis C. (2010) How programming environment shapes perception, learning and goals: logo vs. scratch *SIGCSE '10: Proceedings of the 41st ACM technical symposium on Computer science education pages 346-350*

Papert, S. (1980) *Mindstorms: Children, Computers and Powerful Ideas* Basic Books Inc. USA

Polya, G. (1945) *How to solve it* Princeton University Press USA

Saez-Lopez, J.M., Roman-Gonzalez, M., Vazquez-Cano, E., (2016) Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five Schools *Elsevier computers & Education Volume 97, Pages 129–141*

Scratch (2013-2016) https://scratch.mit.edu

Simon J. (1996) *L'evolució de l'ensenyament del llenguatge logo a l'escola de mestres Blanquerna 10 anys.* VI Seminari Logo Terrassa Spain