Analisi dei processi di pensiero computazionale alla base della creazione di grafici animati per il problem solving

Alice Barana[1[0000-0001-9947-5580]], Cecilia Fissore[1[0000-0001-8398-265X]], Francesco Floris[1[0000-0003-0856-2422]], Marina Marchisio[1[0000-0003-1007-5404]]

[1] Dipartimento di Matematica, Università di Torino, Italia
{alice.barana, cecilia.fissore, francesco.floris, marina.marchisio}@unito.it

Abstract. Tra le competenze chiave di cui tutti gli individui hanno bisogno per la realizzazione professionale, lo sviluppo personale, la cittadinanza attiva, l'inclusione sociale e l'occupazione sono presenti il problem solving e il pensiero computazionale. In matematica, nelle attività di risoluzione di problemi contestualizzati mediante l'utilizzo di un Ambiente di Calcolo Evoluto (ACE), le differenze tra queste due competenze si assottigliano. Un aspetto molto importante di un ACE per la risoluzione dei problemi è la programmazione di grafici animati che permettono di creare un'animazione di un grafico attraverso la generalizzazione di un grafico statico, scegliendo il parametro da far variare e l'intervallo di variazione dello stesso. L'obiettivo di questa ricerca è analizzare i processi di pensiero computazionale alla base della creazione di grafici animati per la risoluzione di un problema contestualizzato. A tal fine sono state selezionale e analizzate alcune risoluzioni di problemi svolte da studenti di classe quarte delle scuole secondarie di secondo grado. Vengono mostrati alcuni esempi in cui sono emersi processi di pensiero computazionale diversi, che riflettono strategie risolutive e processi di generalizzazioni differenti. Dall'analisi è emerso che nella creazione dei grafici animati vengono attivati tutti i processi alla base delle strategie mentali del pensiero computazionale utili per risolvere problemi.

Keywords: Ambiente di Calcolo Evoluto, Grafici animati, Pensiero Computazionale, Problem Solving.

1 Introduzione

Secondo le raccomandazioni del Consiglio dell'Unione Europea del 22 maggio 2018 [1] tra le otto competenze chiave per l'apprendimento permanente per conseguire progressi e successi vi sono la competenza matematica, definita come la "capacità di sviluppare e applicare il pensiero e la comprensione matematici per risolvere una serie di problemi in situazioni quotidiane" e la competenza digitale che "presuppone l'interesse per le tecnologie digitali e il loro utilizzo con dimestichezza e spirito critico e responsabile". Le competenze chiave sono collegate allo sviluppo di capacità quali: la capacità di risoluzione di problemi, il pensiero critico, la capacità di cooperare, la creatività e il

pensiero computazionale. Esse sono alla base di tutte le competenze chiave e consentono di sfruttare in tempo reale ciò che si è appreso, al fine di sviluppare nuove idee, nuove teorie e nuove conoscenze [1].

Il problem solving è un aspetto importante dell'insegnamento e dell'apprendimento della matematica, presente in tutti i curricula matematici [2]. Negli ultimi anni l'uso delle tecnologie digitali nelle attività di problem solving ha consentito una preziosa varietà di rappresentazione ed esplorazione dei compiti matematici [3] offrendo l'opportunità di estendere i modi di ragionare sulle strategie coinvolte nella risoluzione dei problemi [4]. Una delle tecnologie utilizzate per attività di problem solving è un Ambiente di Calcolo Evoluto (ACE) che consente di eseguire calcoli numerici e simbolici, creare rappresentazioni grafiche (statiche e animate) in 2 e 3 dimensioni, scrivere procedure in un linguaggio semplice, programmare e collegare tutti i diversi registri di rappresentazione in un unico foglio di lavoro utilizzando anche il linguaggio verbale [5]. Un aspetto molto importante di un ACE per la risoluzione dei problemi è la progettazione e la programmazione di grafici animati e componenti interattive. I primi permettono di creare un'animazione di un grafico al variare di un parametro, le seconde permettono di visualizzare come variano i risultati, eventualmente anche grafici, di un processo quando vengono modificati i parametri di input.

L'obiettivo di questa ricerca è analizzare i processi di pensiero computazionale alla base della creazione di grafici animati per la risoluzione, attraverso l'utilizzo di un ACE, di un problema contestualizzato. A tal fine verranno analizzati alcuni esempi di risoluzioni di un problema contestualizzato svolte da studenti di classe quarte delle scuole secondarie di secondo grado.

2 Stato dell'arte

L'espressione "pensiero computazionale" (in inglese "computational thinking") è stata resa popolare da un articolo di Jeannette Wing [6] in cui l'autrice sostiene l'importanza di insegnare i concetti fondamentali dell'Informatica nella scuola, possibilmente fin dalle prime classi. Ancora oggi la didattica dell'Informatica, seppur frequentemente venga descritta come lo studio sistematico dei processi computazionali che descrivono e trasformano l'informazione, si riduce nella maggioranza dei casi all'uso dei computer (informatica di consumo) [7].
Inizialmente, Wing [6] non ha dato una definizione precisa del pensiero computazionale ma ne delinea le caratteristiche principali:

- un modo in cui gli esseri umani risolvono un problema;
- basato sul concettualizzare, e non sul programmare, a più livelli di astrazione;
- basato su idee, non artefatti;
- integrazione tra pensiero matematico e ingegneristico;
- per tutti, in tutto il mondo.

Lodi et al. [8] riflettono sul fatto che cercare una definizione operativa precisa e universalmente condivisa dell'espressione, tutt'ora non esistente, può creare confusione

sull'argomento, portando in primo luogo a considerare erroneamente il pensiero computazionale come un nuovo soggetto di insegnamento, concettualmente distinto dall'informatica. Gli autori ritengono più importante utilizzare l'espressione come un modo abbreviato di riferirsi ad un concetto ben strutturato, nucleo fondante dell'Informatica. Nell'articolo descrivono quindi il pensiero computazionale come un processo mentale (o più in generale un modo di pensare) per risolvere problemi (problem solving) e ne definiscono gli elementi costitutivi: strategie mentali, metodi, pratiche e competenze trasversali. Per quanto riguarda le strategie mentali utili per risolvere problemi, gli autori descrivono i seguenti processi mentali:

- pensiero algoritmico: usare il pensiero algoritmico per progettare una sequenza ordinata di passi (istruzioni) per risolvere un problema;
- pensiero logico: usare la logica e il ragionamento per stabilire e controllare i fatti;
- scomposizione di problemi: dividere e modularizzare un problema complesso in semplici sotto-problemi, risolubili in modo più semplice;
- astrazione: liberarsi dei dettagli inutili per concentrarsi sulle idee rilevanti;
- riconoscimento di pattern: individuare regolarità/schemi ricorrenti nei dati e nei problemi;
- generalizzazione: usare le regolarità riconosciute per fare previsioni o per risolvere problemi più generali.

Queste strategie mentali richiamano in molti aspetti le fasi di un'attività di problem solving nella didattica, ad esempio, della Matematica: comprensione del problema, ideazione del modello matematico, risoluzione del modello e interpretazione della soluzione ottenuta [9]. Ciò che distingue il pensiero computazionale dal problem solving è il cambiamento di paradigma concettuale costituito dal passaggio dal risolvere i problemi al far risolvere i problemi [8]. Il primo infatti non riguarda una generica risoluzione di problemi: la formulazione del problema e della soluzione devono essere espresse, scrivendo un algoritmo in un linguaggio opportuno, in modo che un "agente che elabora informazioni" (essere umano o macchina) possa comprendere, interpretare ed eseguire le istruzioni che le vengono fornite. Questa differenza a nostro parere si assottiglia quando vengono proposte attività di problem solving attraverso l'utilizzo di tecnologie e in particolare di ACE. In questo caso, a partire dal pensiero mentale, lo studente deve scegliere come impostare il procedimento risolutivo tra le molteplici modalità possibili (parole, grafici, conti numerici o simbolici, procedure, cicli, etc.) e allo stesso tempo scrivere un algoritmo in un linguaggio opportuno, in modo che l'ACE elabori le informazioni e restituisca un risultato.

In questa ricerca vogliamo concentrarci sulla creazione di grafici animati che comportano la generalizzazione di un grafico statico scegliendo il parametro da far variare e l'intervallo di variazione dello stesso.

Secondo Malara [10], il termine "processo di generalizzazione" comprende "una serie di atti di pensiero che portano un soggetto a riconoscere, esaminando casi singoli, l'occorrenza di elementi caratteristici comuni; a spostare l'attenzione dai singoli casi alla totalità dei casi possibili ed ad estendere a tale totalità i caratteri comuni individuati". Le azioni principali alla base di questo processo sono la riconoscenza di pattern, l'individuazione di somiglianze e il collegamento tra aspetti analoghi. Esse portano il

soggetto a considerare, invece di un singolo caso, tutti i casi possibili e ad estendere e adattare il modello individuato. Questa definizione non è così distante dai processi mentali del pensiero computazionale di riconoscimento di pattern e generalizzazione che abbiamo citato in precedenza. Riguardo al processo di generalizzazione, l'autrice si sofferma su una riflessione di Dörfler [11] che considera cruciale la rappresentazione del processo "attraverso l'uso di oggetti percepibili, come segni scritti, di elementi caratteristici e stage, di passi e risultati delle azioni". In questo modo si genera un protocollo di azioni che permette una ricostruzione e concettualizzazione cognitiva del processo stesso.

3 Creazione di grafici animati con un ACE

La creazione di un grafico animato con un ACE, e in particolare con l'ACE Maple[1], avviene attraverso l'utilizzo del comando "animate" che crea l'animazione di un grafico in 2 o 3 dimensioni al variare di un parametro. Per utilizzare il comando si utilizza la seguente sintassi:

animate(plotcommand, plotargs, t=a..b, options)

dove:

- *plotcommand*: un comando o una procedura Maple che genera un grafico 2D o 3D
- *plotargs*: lista di argomenti del *plotcommand*
- *t*: nome del parametro utilizzato nell'animazione
- *a*, *b*: estremi dell'intervallo di variazione del parametro

Supponiamo, ad esempio, di voler creare un'animazione per visualizzare come varia la concavità di una parabola con vertice nell'origine nel piano cartesiano. Pima di tutto creiamo il grafico statico di una determinata parabola, ad esempio $y = 3x^2$, attraverso il seguente comando:

plot($3x^2$,x=-15..15,color=blue)

Dando invio al comando si ottiene in output il seguente grafico:

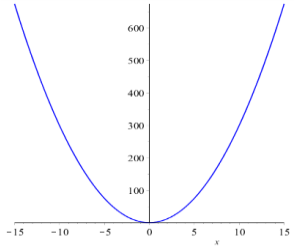

Fig. 1. Grafico statico della parabola $y = 3x^2$

In questo modo vediamo che la concavità della parabola è positiva. Lo stesso procedimento possiamo farlo per disegnare la parabola $y = -3x^2$, disegnando così una parabola con la concavità rivolta verso il basso. Supponiamo ora di voler visualizzare come varia con continuità la concavità della parabola tra la prima e la seconda situazione

[1] https://www.maplesoft.com/

attraverso un'animazione. Utilizziamo quindi il comando animate, tenendo conto che nel comando (1) il *plotcommand* è *plot* e tutto quello che è presente all'interno delle parentesi tonde è il *plotargs*. All'interno del comando *animate* il *plotargs* non può essere lo stesso del grafico statico ma è necessario generalizzarlo riconoscendo pattern comuni tra i grafici della prima e seconda situazione che vogliamo unire in un unico comando. In particolare, scegliamo il parametro da far variare, in questo caso il coefficiente di x^2, e l'intervallo in cui farlo variare, in questo caso l'intervallo dei numeri reali compresi tra 3 e -3. Otteniamo quindi il seguente comando:

$$animate(plot, [a \cdot x^2, x = 15..15, color = blue], a = 3..-3)$$

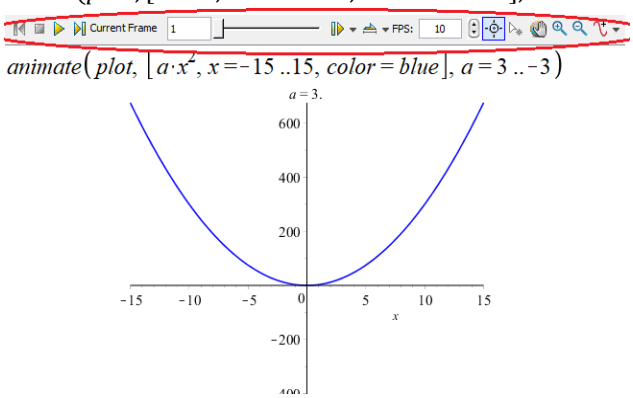

Fig. 2. Grafico animato della parabola $y = a \cdot x^2$, con a che varia da 3 a -3

In questo modo abbiamo creato l'animazione desiderata (Fig. 2) avendo un feedback immediato del risultato della generalizzazione che abbiamo effettuato. Un ulteriore aspetto dell'utilizzo di un ACE consiste nel fatto che attraverso uno specifico linguaggio di programmazione è possibile definire nuovi comandi, chiamati "procedure". All'interno del comando animate è possibile generalizzare una qualsiasi procedura che restituisca in output un grafico in due o tre dimensioni.

Alla luce del quadro teorico studiato, riteniamo che la creazione dell'animazione di un grafico utilizzando un ACE sia un processo di pensiero computazionale e la visualizzazione immediata in output del risultato della generalizzazione effettuata possa aiutare fortemente gli studenti nello sviluppo di questa competenza.

4 Metodologia

Per analizzare i processi di pensiero computazionale alla base della creazione di grafici animati abbiamo analizzato la risoluzione, mediante l'utilizzo di un ACE, di un problema contestualizzato effettuata da studenti di classe quarta della scuola secondaria di secondo grado nell'ambito del Progetto Digital Math Training [12, 13].

Il problema, intitolato "Coccinella", parla di una coccinella posata sulla ruota posteriore di una bicicletta, sulla parte del cerchione più vicina al terreno. Pensando alla ruota della bici come una circonferenza centrata nell'origine, la posizione della coccinella

corrisponde all'angolo di ampiezza $\frac{3}{2}\pi$. Le ruote della bici hanno diametro totale (inclusa la camera d'aria) pari a 60 centimetri, mentre la camera d'aria è spessa 4 centimetri. La richiesta del problema sulla quale ci vogliamo focalizzare riguarda la traiettoria percorsa dalla coccinella in un chilometro, immaginando che non si sia mai mossa. La traiettoria compiuta dalla coccinella è data dalla combinazione di 2 moti: quello circolare della ruota e quello rettilineo della bici. Essa è quindi rappresentata da una cicloide, il cui grafico si può certamente considerare una risposta corretta alla richiesta del problema. Tuttavia, il grafico statico non mostra come si sia arrivati a quel risultato e il procedimento effettuato per arrivare alla soluzione. Invece attraverso un grafico animato è possibile visualizzare il moto della ruota e la traiettoria percorsa dalla coccinella durante il moto stesso. Inoltre, è possibile tenere traccia dei frames dell'animazione e in questo modo viene tracciata direttamente la traiettoria cercata.

Per la nostra ricerca abbiamo visionato 80 soluzioni proposte da altrettanti studenti, tra le quali abbiamo selezionato e analizzato le 16 in cui è stata utilizzata un'animazione per visualizzare la traiettoria della coccinella.

5 Risultati

In tutte le 16 soluzioni analizzate in cui è stata utilizzata un'animazione si evidenzia l'utilizzo del pensiero algoritmico per costruire il grafico animato come risultato di una sequenza ordinata di comandi. Tuttavia, sono emersi anche processi di pensiero computazionale diversi, in alcuni casi vincenti e in altri meno, che riflettono strategie risolutive differenti. Abbiamo classificato le soluzioni analizzate in quattro categorie:

1. animazione come creazione di una curva punto per punto (2 soluzioni);
2. animazione come verifica del risultato ottenuto (3 soluzioni);
3. animazione come unione di animazioni (5 soluzioni);
4. animazione di una procedura (1 soluzione);
5. animazioni con procedimenti errati (5 soluzioni).

Sebbene le soluzioni classificate all'interno di una stessa categoria non siano identiche, esse mostrano analoghi processi di pensiero computazionale (presentati nel quadro teorico). Per ogni categoria sopracitata verrà quindi presentato un esempio di risoluzione paradigmatico.

Nelle soluzioni della prima categoria, il grafico della cicloide è stato ottenuto attraverso il comando *animatecurve* che serve a creare l'animazione del disegno di una curva in 2 dimensioni, dando in input le equazioni parametriche della curva che definiscono le coordinate dei suoi punti al variare di un parametro. La sintassi utilizzata in questo esempio è stata la seguente:

animatecurve([30 sin(t) + 30 t, 30 + 30 cos(t), t = 0 .. 10 Pi], color = black, frames = 100, thickness = 2, title = "Moto della coccinella sul cerchione")

In output è stato ottenuto il seguente grafico che rappresenta la creazione della curva punto per punto:

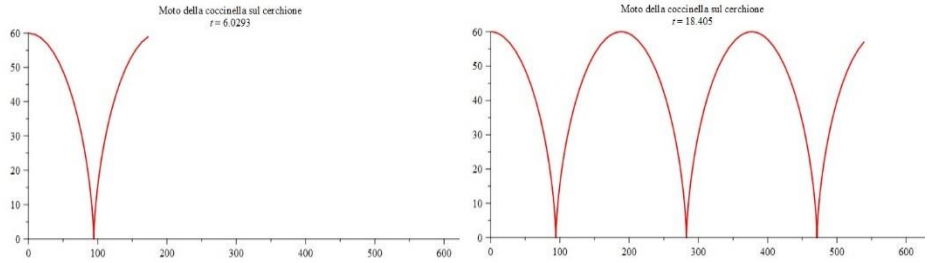

Fig. 3. Esempio di animazione come creazione di una curva punto per punto.

Per ricavare le equazioni parametriche della curva sono stati studiati i moti di traslazione e rotazione della ruota, utilizzando il processo mentale della scomposizione di problemi. Dopodiché è stato utilizzato il processo di generalizzazione e astrazione per unire i due moti e ottenere, con il comando *animate*, la traiettoria della coccinella calcolando la sua posizione puntuale al variare del tempo.
La generalizzazione e l'utilizzo del comando in questo caso sono stati effettuati correttamente, ma non è corretto il procedimento risolutivo perché è stato considerato un errato punto di partenza.

Nella seconda categoria abbiamo incluso le soluzioni in cui è stato ricavato il grafico statico della cicloide e poi è stato animato un punto che si muove su di essa. Nell'esempio che presentiamo, il grafico della cicloide è stato ottenuto utilizzando la forma parametrica della curva con il seguente comando:

graf:=plot([0.3(t-sin(t)),0.3(1-cos(t)), t = 0 .. 8Pi], color = black)

Il comando per creare l'animazione è il seguente:

animate(pointplot, [[0.3 (t-sin(t)), 0.3(1-cos(t))], color = red, symbol = solidcircle, legend = "coccinella"], t = 0 .. 8Pi, frames = 50, background = graf)

Si ottiene l'animazione rappresentata nella seguente figura:

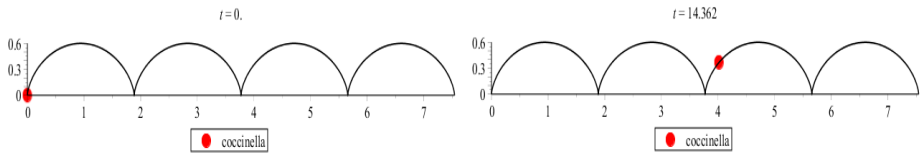

Fig. 4. Esempio di animazione come verifica del grafico statico ottenuto

Questo procedimento è stato utilizzato per verificare il grafico statico ottenuto osservando direttamente come si muove la coccinella. In questo caso il processo di generalizzazione riguarda esclusivamente il grafico del punto e di conseguenza le sue coordinate. Il grafico della cicloide è stato utilizzato come sfondo dell'animazione. Il processo di astrazione è avvenuto in due fasi: nella prima fase è stata effettuata una generalizzazione per creare un comando statico (la curva parametrica), nella seconda fase è stata sfruttata la generalizzazione per creare l'animazione. Alla base della creazione di questa animazione c'è il processo logico per controllare la correttezza dei risultati precedentemente ottenuti.

La terza categoria riguarda le soluzioni in cui è stato presentato il grafico animato contenente: le due circonferenze per rappresentare la ruota, il centro della ruota, la coccinella e il raggio della bicicletta che unisce la coccinella al centro della ruota. Per farlo sono stati creati 5 grafici animati diversi e sono stati poi visualizzati in un'unica finestra grafica attraverso il comando *display*. Nel seguente esempio, sono stati prima definite le tre procedure *centro, coccinel, R* per rappresentare in modo statico rispettivamente la coccinella, il raggio e il centro della ruota. Dopodiché sono stati animati singolarmente i grafici di questi tre elementi e delle due circonferenze per rappresentare le ruote e infine sono stati unite tutte le animazioni in un unico grafico attraverso il seguente comando:

display({animate(R, [t],t=0 ..8Pi, scaling = constrained, frames = 80), animate(centro, [3t, 3], t = 0 .. 8Pi, frames=80), animate(implicitplot, [x^2+y^2+9t^2-6tx-6y+2.24 = 0, x=0..80, y=0..8, color = black], t=0..8Pi, frames = 80), animate(implicitplot, [9t^2-6tx+x^2+y^2-6y = 0, x=0..80, y=0..10, color = black], t=0 .. 8Pi, frames = 80), animate(coccinel, [3t-2.6sin(t), 3-2.6cos(t)], t = 0..8Pi, frames = 80, trace = 90000)}, labels = ["x (dm)", "y (dm)"])

In questo comando è stata utilizzata l'opzione trace per i grafici animati che permette di selezionare un insieme di fotogrammi da stampare mentre viene eseguita l'animazione, tenendone così traccia. L'output del comando è rappresentato nella seguente figura:

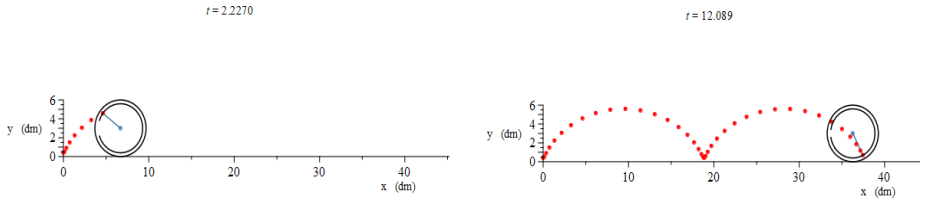

Fig. 5. Esempio di animazione come unione di animazioni

In questa strategia risolutiva, utilizzata dalla maggior parte degli studenti, per creare l'animazione è stato utilizzato il processo mentale della scomposizione di problemi: è stata divisa un'animazione complessa in diverse animazioni, risolubili in modo più semplice. L'unione delle varie animazioni è stata possibile dal momento che sono state effettuate tutte in base allo stesso parametro con lo stesso intervallo di variazione, utilizzando il processo di riconoscimento di pattern e di generalizzazione.

L'ultima soluzione analizzata, anche se dal punto di vista matematico non del tutto corretta, è molto interessante dal punto di vista del pensiero computazionale messo in atto per realizzarla. Essa consiste nella creazione di un'animazione in cui sulla sinistra del grafico viene rappresentata la coccinella che si muove lungo la ruota e sulla destra viene rappresentata la traiettoria, tracciata punto per punto con un segmento (Fig 7). Per creare il grafico, diversamente dall'esempio precedente, non sono state unite diverse animazioni bensì è stata definita una procedura per unire diversi comandi grafici al variare dello stesso parametro, ed è stata poi animata. La procedura definita è la seguente:

*F := **proc** (t) plots[display](plottools[line]([-2, 0], [sin(t)-2, -cos(t)], color = blue), plottools[line]([sin(t)-2, -cos(t)], [t, abs(2sin((1/2)t))], color = blue), plot(abs(2sin((1/2)x)), x = 0 .. t, color = "Green", legend = ["Traiettoria"])) **end proc***
Questa procedura, dato in input il valore del parametro *t*, restituisce il grafico statico del raggio della ruota, il grafico della curva da 0 fino a *t* e il segmento che unisce la posizione della coccinella sulla ruota a quella sulla traiettoria. Ad esempio, per il valore *t=3,14* si ottiene il seguente grafico:

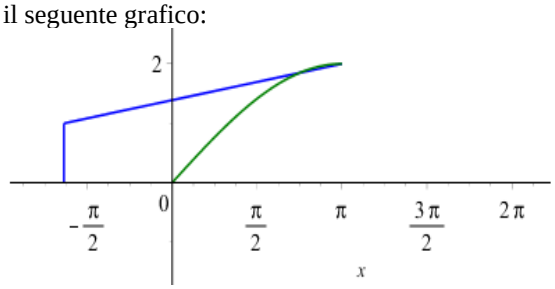

Fig. 6. Grafico statico per la costruzione dell'animazione di una procedura

Dopodiché è stato utilizzato il comando animate per animare la procedura al variare del parametro di input, aggiungendo come sfondo il grafico statico della circonferenza che rappresenta la ruota. Il risultato è rappresentato nella seguente figura:

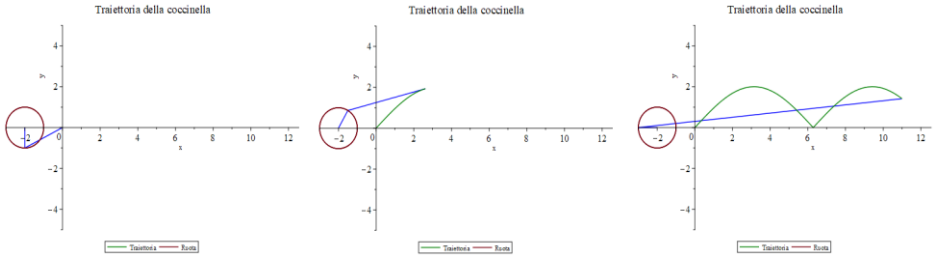

Fig. 7. Esempio di animazione di una procedura

La strategia risolutiva utilizzata è diversa da quella dall'esempio precedente perché invece di unire animazioni differenti in un unico grafico animato, l'animazione è come se venisse creata come una sequenza ravvicinata di diverse istantanee dello stesso moto. Dal punto di vista del pensiero computazionale questo richiede un processo di astrazione maggiore in quanto il processo di generalizzazione attraverso il riconoscimento di pattern viene effettuato due volte: una per definire la procedura e una per animarla.

Le restanti 5 soluzioni analizzate contenevano animazioni errate per diversi aspetti: l'utilizzo del comando (errato processo di generalizzazione), la scala scelta per visualizzare il grafico (errato processo logico) o la contestualizzazione del grafico (visualizzato nel semiasse negativo delle ordinate, frutto di un errato processo di astrazione). Questo conferma che tutti gli aspetti del pensiero computazionale, così come del problem solving, sono fondamentali.

6 Conclusioni

I risultati mostrano alcuni esempi di processi di pensiero computazionale nella creazione di grafici animati per la risoluzione di un problema contestualizzato attraverso l'utilizzo di un ACE. Analizzando le animazioni create per ricavare la traiettoria della coccinella sono emersi quattro processi di pensiero computazionale diversi, che riflettono strategie risolutive e processi di generalizzazioni differenti. Dai risultati è emerso che nella creazione dei grafici animati vengono attivati tutti i processi alla base delle strategie mentali del pensiero computazionale utili per risolvere problemi. È auspicabile quindi, anche per la loro componente ludica, che vengano usati nella didattica ordinaria della matematica.

References

1. Commissione Europea: Raccomandazione del Consiglio, del 22 maggio 2018, relativa alle competenze chiave per l'apprendimento permanente. Official Journal of the European Union. 1–13 (2018).
2. Liljedahl, P., Santos-Trigo, M., Malaspina, U., Bruder, R.: Problem solving in mathematics education. Springer Berlin Heidelberg, New York, NY (2016).
3. Santos-Trigo, M., Moreno-Armella, L., Camacho-Machín, M.: Problem solving and the use of digital technologies within the Mathematical Working Space framework. ZDM. 48, 827–842 (2016). https://doi.org/10.1007/s11858-016-0757-0.
4. Kuzniak, A., Parzysz, B., Vivier, L.: Trajectory of a problem: a study in Teacher Training. 10, 407–440 (2013).
5. Barana, A., Fioravera, M., Marchisio, M.: Developing problem solving competences through the resolution of contextualized problems with an Advanced Computing Environment. In: Proceedings of the 3rd International Conference on Higher Education Advances. Universitat Politècnica València (2017). https://doi.org/10.4995/HEAD17.2017.5505.
6. Wing, J.: Computational thinking. Presented at the Communications of the ACM (2006).
7. Bizzarri, G., Forlizzi, L., Proietti, G.: Informatica: didattica possibile e pensiero computazionale. 10 (2011).
8. Lodi, M., Martini, S., Nardelli, E.: Abbiamo davvero bisogno del pensiero computazionale? 15 (2017).
9. Samo, D.D., Darhim, D., Kartasasmita, B.: Culture-Based Contextual Learning to Increase Problem-Solving Ability of First Year University Student. Journal on Mathematics Education. 9, (2017). https://doi.org/10.22342/jme.9.1.4125.81-94.
10. Malara, N.A.: Processi di generalizzazione nell'insegnamento/apprendimento dell'algebra. Annali online formazione docente. 4, 13–35 (2013).
11. Dorfler, W.: Forms and means of generalization in mathematics. In: Mathematical Knowledge: Its growth through teaching. pp. 63–95. Kluver (1991).
12. Barana, A., Marchisio, M.: Sviluppare competenze di problem solving e di collaborative working nell'alternanza scuola-lavoro attraverso il Digital Mate Training. In: Atti di Didamatica. pp. 1–10 (2017).
13. Barana, A., Marchisio, M.: Dall'esperienza di Digital Mate Training all'attività di Alternanza Scuola Lavoro. MONDO DIGITALE. 15, 10 (2016).