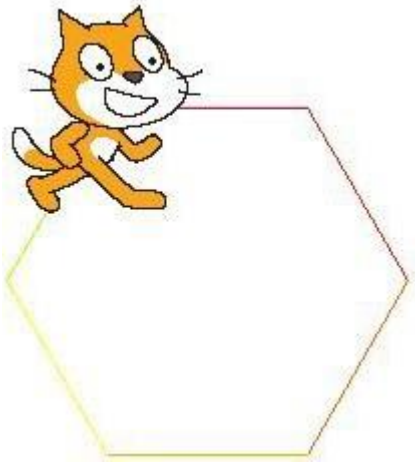
Scratch, imparare divertendosi



Prof.ssa Sabrina Fenu

Università Roma 3 - Luglio 2014

Scratch, imparare divertendosi

Questo lavoro propone un percorso di Coding mediante Scratch (versione 1.4) basato su una serie di esercitazioni di livello base con difficoltà progressiva. Tale percorso si basa sull'acquisizione di competenze logiche, matematiche, digitali in maniera creativa, creando attraverso lo spirito ludico del gioco dei Logo un interesse verso materie tecniche spesso più prescrittive e strutturate.
Scratch è un software open source che si adatta a vari contesti, da quello didattico generale a quello di recupero impostato in situazioni di alto grado di dispersione e le Unità didattiche proposte possono essere introdotte in un qualsiasi percorso base di Coding.
Il lavoro è strutturato in una prima parte introduttiva, con la quale attraverso semplici riflessioni personali si paragona lo sviluppo computazionale alla programmazione cognitiva, [2] intesa come carico cognitivo da sviluppare, dove perapprofondimenti si rimanda a "la charge cognitive[1] " di Sweller, un libro da valutare sia per le riflessioni relative alla programmazione cognitiva, sia per la risultante fortemente inclusiva.
Nella seconda parte vi sono le Unità didattiche complete di teoria, pratica, esercitazioni guidate e verifiche da proporre, proposte per difficoltà progressiva.

Sabrina Fenu

1 Per ulteriori approfondimenti si rimanda a "la Charge Cognitive" di Sweller e Tricot.
2 Scratch," imparare divertendosi" è la rielaborazione del proprio personale lavoro di tesi di abilitazione" Scratch, introduzione alla Programmazione cognitiva, Giugno 2014; il capitolo introduttivo è stato preso da tale lavoro.

Indice generale

Introduzione

In questo lavoro viene affrontato, in una prima parte Scratch come approccio psicopedagogico evidenziando un'importante legame come sviluppo del carico cognitivo contenuto in ognuno di noi. In secondo luogo vi è la sua fluenza digitale secondo la quale si adatta fortemente ai giovani di oggi così definiti da M.Resnick "nativi digitali".
Nella prima si parla proprio di introduzione di Sprite, Stage, Sfondi e scenari e Costumi e della loro modifica. Nell'Unità "primi elementi di programmazione" invece con esempi semplici matematici e non, si cerca diaffrontare i concetti di variabili, assegnazioni, selezione, cicli e ripetizioni.
Nella seconda parte si parla di primi elementi di programmazione introducendo i contenuti e le Unità didattiche complete di esercitazioni e verifiche relative a : Sprite, Sfondi e Costumi, e all'Unità didattica "primi elementi di programmazione".

1.1 Scratch e la carica cognitiva

La lettura di un libro "la charge cognitive" mi ha condotto ad alcune riflessioni che ritengo possono legarsi con l'utilizzo di Scratch come "linguaggio cognitivo". Tricot, Chanquoy e Sweller sostengono che per facilitare la comprensione di certe teorie è fondamentale pratica ed esperienza.
Si parla di vari tipi di memoria: breve termine, lungo termine, dichiarativa, procedurale, esplicita, implicita, visiva, memoria da lavoro..
Scratch è un' ottimo strumento di apprendimento grazie allo sviluppo di determinate abilità che si legano fortemente alla memoria visiva, implicita, esplicita e a lungo termine. Questo perché con Scratch si impara facendo. Un vecchio proverbio cinese le cui origini sembra siano fatte risalire a Confucio, poi tradotto da Munari citava questo" Seascolto dimentico, se vedo imparo, se faccio capisco".
Ebbene Scratch si presta ad essere imparato con la pratica, una pratica semplice, divertente, intuitiva, cognitiva.
Vorrei parlare di Scratch anche come linguaggio cognitivo, e vi spiego perché. John Sweller esprime la teoria dei fondamenti della carica cognitiva, cosa è, come funziona, quali conseguenze ha, come svilupparla nella vita di ogni giorno, nella vita professionale, nella scuola.

Ma il nucleo centrale è come sviluppare l'apprendimento e il lavoro dell'alunno. Tricot parla di memoria a breve termine e a lungo termine, memoria visiva, memoria da lavoro... Io credo che una buona parte di esse si leghino fortemente all'apprendimento mediante un linguaggio di programmazione che insegna divertendo, e soprattutto "facendo", perchè si sviluppa la memoria di attenzione, i processi emotivi, la socializzazione.
Parlo di Socializzazione perché Scratch è usato in più di 150 paesi ed è disponibile in più di 40 lingue diverse. Le lingue vengono inserite con l'aiuto degli utenti che possono inserire una nuova lingua o migliorare una traduzione.
Ogni nuovo progetto realizzato può essere condiviso sulla piattaforma che attualmente ospita quasi 4.700 progetti.
Scratch è social infatti si può trovare il sito di Scratch al seguente indirizzo http://scratch.mit.edu/e cliccando su crea si può cominciare subito ad utilizzarlo.
Scratch si presta alla memoria a lungo termine, perché il suo impatto grafico esplicativo attraverso i suoi blocchi è difficile da dimenticare, come del resto è difficile dimenticare la costruzione mediante i Lego.
Io credo che questo genere di tecnologia istruttiva sia fondamentale nella crescita personale e perché no professionale del ragazzo sin da bambino.
Ogni programmatore affronta diverse fasi per dare vita ad un progetto: l'elaborazione dell'idea, la creazione di un prototipo funzionale, il test pratico, la correzione del progetto se i risultati non sono quelli attesi, l'ottenimento di un feedback da altri, la revisione e la riprogettazione.
Questo processo di progettazione è sicuramente uno stimolo per lo sviluppo della creatività dei ragazzi, delle loro capacità comunicative, che li aiuta a selezionare ciò che è utile in schemi mentali organizzati.
Il motto di Scratch è "Digital fluency" (Fluidità digitale) che significa progettazione e creazione e non solo saper navigare, chattare e interagire. E come diceva il suo creatore Resnick, Scratch si adatta ai ragazzi di oggi, definiti "nativi digitali" in quanto i giovani di oggi usano i giochi iterattivi, le animazioni, le simulazioni, usano ma non creano.
Non sono abituati a creare, e quindi a ragionare e sviluppare nel lungo periodo, loro hanno bisogno di vedere l'immediatezza ed è per questo che Scratch si adatta a questa generazione
Le abilità che Scratch sviluppa sono valide per altre discipline, logiche, matematiche, fisiche, di apprendimento della lingua inglese.
Sarebbe una crescita anche per noi docenti introdurre Scratch nella nostra programmazione annuale in maniera importante, sviluppando magari anche la

Parte con interfaccia hardware (vedi Arduino) che fa vedere la gestione Robot, e la programmazione più avanzata delle liste con BYOB.

1.2 Scratch contro la dispersione

L'educazione è una cosa ammirevole, ma è bene ricordare, di tanto in tanto, che nulla che valga la pena di conoscere si può insegnare. (Oscar Wilde)
Gli studi mostrano come la programmazione introdotta a scuola possa influire positivamente sulle capacità cognitive degli alunni. Si ritiene che, per la buona riuscita di un "corso di programmazione", sia molto importante l'aspetto ludico e il ruolo dell'insegnante, che, per primo, deve amare ciò che insegna e deve sapere motivare gli alunni nell'apprendimento. Il binomio fra questi due elementi rappresenta il primo passo per una buona riuscita nella didattica.

1.3 Divertimento per imparare meglio

Nei primi anni l'educazione sia una specie di divertimento; vi sarà così più facile scoprire le inclinazioni naturali (Platone)

È importante che, durante l'apprendimento, l'alunno si diverta e lo veda come se fosse un gioco perché questo ha come effetto l'incremento delle conoscenze, delle abilità, delle competenze e, contemporaneamente, esercita le funzioni cognitive. Vedere le attività proposte come un gioco, e non come una lezione tradizionale, porta ad un maggiore entusiasmo e coinvolgimento degli alunni. In questo modo avviene un potenziamento cognitivo e si perseguono gli obiettivi didattici, che però rimangono nascosti al discente.

Anche le attività progettate con i migliori obiettivi e presupposti non hanno valore se non riescono a coinvolgere emotivamente e cognitivamente l'alunno. È quindi necessario, per la riuscita della didattica, offrire attività divertenti e/o ludiche che, mentre sono attraenti, educano dal punto di vista cognitivo, relazionale e materiale. Un insegnante che riesce a cogliere quest'aspetto può

favorire lo sviluppo di diversi aspetti cognitivi. Il gioco e il divertimento permettono anche di affrontare serenamente situazioni problematiche. Ad esempio un alunno che ha il timore di non riuscire a svolgere un compito assegnato, può superare quest'inibizione se viene messo in una situazione di gioco, ancor meglio se il gioco è di gruppo e non prevede una valutazione.
Per riuscire a far breccia nella "percezione di inefficacia" del ragazzo, è bene proporre inizialmente compiti abbastanza facili, in modo tale da rafforzare la propria percezione di efficacia, per poi passare gradualmente ad attività più complesse che inducono un sentimento di "sfida" e sono maggiormente stimolanti.

1.4 Scratch, non solo un linguaggio di programmazione

Scratch è un linguaggio di programmazione open source e multipiattaforma, presentato per la prima volta nel 2007, che permette di creare facilmente storie interattive, giochi, animazioni e di condividerli sul web. È stato creato da M.Resnick e dal suo team al MIT con l'obiettivo di avvicinare i giovani alla programmazione, di far capire loro la logica degli algoritmi e di sviluppare abilità creative nell'uso dei computer. Il suo nome deriva dalla tecnica dei disk jockey che, negli anni '80, introdussero una nuova pratica, lo scratching, che consisteva nel mixare i dischi facendoli ruotare con le mani. La sua semplicità e il suo successo è dovuta al poter sviluppare programmi semplicemente trascinando e combinando tra di loro gli oggetti presenti nel menù. Questo facilita la comprensione della programmazione agli alunni di tutte le età. Un aspetto innovativo, che incuriosisce e che stimola i ragazzi, è che viene favorita l'interazione tra i "programmatori": sul sito ufficiale si possono scambiare pareri, condividere il proprio lavoro e ricevere feedback per le proprie creazioni. Questo passo è molto importante perché permette a persone distanti fisicamente di comunicare, di apprendere le une dalle altre e crea nuove forme di conoscenza. Gli autori rivolgono il linguaggio ai bambini dagli 8 anni in su, ma non mancano esperienze fatte con alunni di età inferiore e con studenti del college.
Programmare con Scratch è come giocare con i LEGO perché le istruzioni sono dei blocchi colorati simili a dei mattoncini e possono essere associati, impilati, uniti e divisi, facendo di ogni progetto qualcosa di originale, come per quelli fatti con i LEGO.

1.5 Cenni sui linguaggi di programmazione

In informatica, un **linguaggio di programmazione** è un linguaggio formale, dotato (al pari di un qualsiasi linguaggio naturale) di un lessico, di una sintassi e di una semantica ben definiti, utilizzabile per il controllo del comportamento di una macchina formale o di una implementazione di essa (tipicamente, un computer).
Il primo linguaggio di programmazione della storia, se si esclude il linguaggio meccanico adoperato da Ada Lovelace per la programmazione della macchina di Charles Babbage, è a rigor di termini il Plankalkül di Konrad Zuse, sviluppato da lui nella Svizzera neutrale durante la seconda guerra mondiale e pubblicato nel 1946. Plankalkül non venne mai realmente usato per programmare. La programmazione dei primi elaboratori veniva fatta invece in short code da cui poi si è evoluto l'assembly, che costituisce una rappresentazione simbolica del linguaggio macchina. La sola forma di controllo di flusso è l'istruzione di salto condizionato, che porta a scrivere programmi molto difficili da seguire logicamente per via dei continui salti da un punto all'altro del codice. La maggior parte dei linguaggi di programmazione successivi cercarono di astrarsi da tale livello basilare, dando la possibilità di rappresentare strutture dati e strutture di controllo più generali e più vicine alla maniera (umana) di rappresentare i termini dei problemi per i quali ci si prefigge di scrivere programmi.
Tra i primi linguaggi ad alto livello a raggiungere una certa popolarità ci fu il Fortran, creato nel 1957 da John Backus, da cui derivò successivamente il BASIC (1964).
Oltre al salto condizionato, reso con l'istruzione IF, questa nuova generazione di linguaggi introduce nuove strutture di controllo di flusso come i cicli WHILE e FOR e le istruzioni CASE e SWITCH: in questo modo diminuisce molto il ricorso alle istruzioni di salto (GOTO), cosa che rende il codice più chiaro ed elegante, e quindi di più facile manutenzione. Dopo la comparsa del Fortran nacquero una serie di altri linguaggi di programmazione storici, che implementarono una serie di idee e paradigmi innovativi: i più importanti sono l'ALGOL (1960) e il Lisp (1959). Tutti i linguaggi di programmazione oggi esistenti possono essere considerati discendenti da uno o più di questi primi linguaggi, di cui mutuano molti concetti di base; l'ultimo grande progenitore dei linguaggi moderni fu il Simula (1967), che introdusse per primo il concetto (allora appena abbozzato) di *oggetto* software. Nel 1970 Niklaus Wirth pubblica il Pascal, il primo linguaggio strutturato, a scopo didattico; nel 1972 dal BCPL nascono prima il B (rapidamente dimenticato) e poi il C, che invece fu

fin dall'inizio un grande successo. Nello stesso anno compare anche il Prolog, finora il principale esempio di linguaggio logico, che pur non essendo di norma utilizzato per lo sviluppo industriale del software (a causa della sua inefficienza) rappresenta una possibilità teorica estremamente affascinante. Con i primi mini e microcomputer e le ricerche a Palo Alto, nel 1983 vede la luce Smalltalk, il primo linguaggio realmente e completamente ad oggetti, che si ispira al Simula e al Lisp: oltre a essere in uso tutt'oggi in determinati settori, Smalltalk viene ricordato per l'influenza enorme che ha esercitato sulla storia dei linguaggi di programmazione, introducendo il paradigma object-oriented nella sua prima incarnazione *matura*. Esempi di linguaggi object-oriented odierni sono Eiffel (1986), C++ (che esce nello stesso anno di Eiffel) e successivamente Java, classe 1995.

Non ha senso, in generale, parlare di linguaggi migliori o peggiori, o di linguaggi migliori in assoluto: ogni linguaggio nasce per affrontare una classe di problemi più o meno ampia, in un certo modo e in un certo ambito. Però, dovendo dire se un dato linguaggio sia adatto o no per un certo uso, è necessario valutare le caratteristiche dei vari linguaggi. Distinguiamo tra due tipi di caratteristiche principali: le caratteristiche intrinseche, e quelle esterne. Tra le caratteristiche intrinseche: espressività, leggibilità, generalità, robustezza, modularità, flessibilità, efficienza e coerenza, quella che interessa di più a noi e' la **didattica.**

Questa caratteristica indica la semplicità del linguaggio e la rapidità con cui lo si può imparare. Il BASIC, per esempio, è un linguaggio facile da imparare: poche regole, una sintassi molto chiara e limiti ben definiti fra quello che è permesso e quello che non lo è. Il Pascal non solo ha i pregi del BASIC ma educa anche il neo-programmatore ad adottare uno stile corretto che evita molti errori e porta a scrivere codice migliore. Al contrario, il C non è un linguaggio didattico perché pur avendo poche regole ha una semantica molto complessa, a volte oscura, che lo rende molto efficiente ed espressivo ma richiede tempo per essere padroneggiata.

Negli ultimi anni, i dipartimenti di informatica presso diverse università stanno compiendo sforzi per attirare un maggiore numero di studenti. Alcuni paesi, ad esempio gli Stati Uniti d'America, hanno dichiarato che risolvere la carenza di studenti in campo informatico è una priorità nazionale. Uno dei metodi principali di questo sforzo è l'abbandono di un approccio eccessivamente improntato sulle tecniche di programmazione e la ricerca di un linguaggio che eviti la complessità sintattica.

Il Logo è un linguaggio di programmazione per computer creato da Seymour

Papert nel 1967 all'interno del MIT (Massachusetts Institute of Tecnology). Esso è stato pensato come strumento per agevolare e migliorare l'apprendimento. Le sue caratteristiche possono essere riassunte nelle espressioni: modularità, estensibilità, interattività e flessibilità.
La programmazione con il Logo infatti non è mai fine a se stessa, ma sempre pensata in relazione ad un progetto relativo, alle discipline più svariate: alla matematica, alla lingua, alla musica, alla realizzazione di un videogioco o di un robot... Fra gli ambienti di apprendimento che il Logo offre, quello più conosciuto e sicuramente più usato nelle scuole, anche in Italia, è la Geometria della Tartaruga. Originariamente la Tartaruga era un robot che si muoveva su una superficie tramite comandi impartiti attraverso un computer.
In seguito divenne uno strumento grafico, fu trasferita sul monitor del computer ed usata per disegnare e creare immagini. Nel corso degli anni '70 il Logo ha cominciato a diffondersi e sono state sviluppate diverse esperienze soprattutto negli Stati Uniti; nel 1980 è nata la LCSI (Logo Computer Systems Inc.), che crea, implementa e produce le varie versioni di Logo che conosciamo anche in Italia. Nello stesso anno Seymour Papert pubblica Mindstorms, tradotto in Italia con il sottotitolo: "Ali per la mente", un testo in cui non soltanto l'autore ha spiegato le caratteristiche e le potenzialità del Logo, ma ha anche delineato le idee portanti di un tipo di pedagogia che assieme ad altri modelli teorici è compresa sotto il nome di costruttivismo.
Nel corso degli anni '80 l'uso del Logo è cresciuto considerevolmente e sono state realizzate molte esperienze anche in Italia. Negli Stati Uniti intanto, alla LCSI viene introdotto il Logo Writer, che oltre al linguaggio di programmazione include un word processor ed un'interfaccia semplificata e più intuitiva. Un'altra innovazione di quegli anni è costituita da LEGOLogo. E' stato cioè ideato un sistema che usa il Logo come interfaccia per motori, luci e sensori incorporati nelle macchine costruite dalla LEGO.
Questo esperimento è diventato negli Stati Uniti un grande successo commerciale che ha richiamato l'attenzione di molti insegnanti e studenti.
Ora ci concentriamo su due linguaggi che sono stati influenzati da LOGO e che ne hanno migliorato l'aspetto grafico, rendendolo ancora piu' accessibile ai ragazzi di qualsiasi età, compresi i più piccoli: Scratrch e B.Y.O.B.

1.6 Studi su Scratch

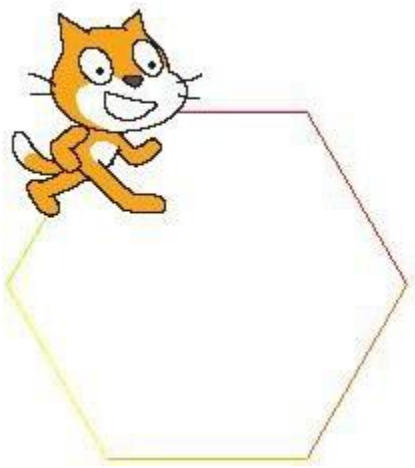
Scratch è un linguaggio nuovo e non sono stati condotti ancora molti studi su esso. Uno studio condotto dal National Center for Women & Information Technology (NCWIT) [31], ha definito Scratch una pratica utile a ridurre le

differenze di sesso nelle tecnologie. Lo studio evidenzia che l'utilizzo è importante per un apprendimento attivo e per permettere ai giovani programmatori di esprimere facilmente la loro creatività. Scratch viene usato per incentivare i giovani a programmare perché, con la sua semplicità, permette di ridurre le barriere date dai linguaggi di programmazione che sono astratti e hanno una sintassi complessa. Lo studio condotto da David J. Malan e Henry H. Leitner ha mostrato che, l'utilizzo di Scratch come linguaggio di programmazione, può attirare un maggiore numero di studenti a studiare informatica; un esempio è dato dall'introduzione di Scratch ad Harvard che ha portato a un calo degli studenti che abbandonavano gli studi, a una diminuzione dei voti negativi e ad un incremento delle studentesse.
Scratch ha avuto molto successo tra i giovani della comunità del Computer Clubhouse46: i ragazzi hanno esplorato le potenzialità del programma scoprendo i concetti di interazione con l'utente, dei condizionali, della comunicazione, della sincronizzazione e dei numeri casuali. Questo ha sorpreso i ricercatori perché i ragazzi non avevano ricevuto della basi di programmazione precedentemente. Un altro dato che ha colpito i ricercatori è che i ragazzi, tra tutti i programmi che avevano a disposizione, sceglievano di usare Scratch. Gli autori pensano che in base alle esperienze che hanno condotto, hanno concluso che la programmazione può essere accessibile anche ai principianti se vengono semplificate le regole e i meccanismi e se vengono forniti supporto e motivazione. Gli studi finora descritti hanno valutato Scratch in situazioni informali, o dopo le attività scolastiche o con studenti adulti, con l'obiettivo di introdurre la programmazione a scuola ad alunni che non la conoscevano. La ricerca si concentra su due possibili obiettivi: affettivi e cognitivi; inoltre, vogliono verificare se i giovani lo trovano divertente da usare nel contesto scolastico e se insegna correttamente i concetti della programmazione. La ricerca viene condotta in una scuola in una zona relativamente povera in una classe di 21 alunni con età compresa tra gli 8 e i 9 anni. Alla fine del percorso, i ragazzi si sono divertiti e hanno valutato positivamente l'esperienza. A livello cognitivo sono avvenuti dei miglioramenti, ma non sono ritenuti significativi perché l'esperienza è stata condotta per un breve periodo (2 mesi). I pregi di Scratch che gli autori riscontrano è che in solo 2 mesi gli alunni hanno potuto affrontare gli elementi chiavedella programmazione.
Il **Computer Clubhouse** è un network mondiale per apprendimento dislocato in 20 stati. È stato fondato da Mitchel Resnik e Natalie Ruskcon del MIT con lo scopo di offrire un ambiente di apprendimento creativo e sicuro dopo la scuola ai giovani meno abbienti.

Cap. 2 Unità Didattica "Sprite, Sfondi, Costumi"
2.1 Il Contesto

Studenti del primo biennio di una Scuola Secondaria di un Istituto Tecnico

18 ore per unità didattica suddivise in: 6h di lezione frontale; 6h di laboratorio; 6h di verifica suddivise in:1h verifica prerequisiti; 1h di verifica teorica con Test;2h di verifica di laboratorio;1h verifica di recupero; 1h correzione verifica



2.2.Prerequisiti

L'alunno prima di affrontare questa Unità Didattica deve aver affrontato:

- Conoscenza di base di matematica
- Conoscenza di primi elementi di programmazione
- Concetti sul diagramma di flusso
- Conoscenza dell'interfaccia di scratch
- Conoscenza del significato sommario dei blocchetti istruzione
- Conoscenza sommaria dei cicli e delle condizioni

2.3 Conoscenze, Competenze e abilità

Conoscenze

Al termine dell'Unità Didattica lo studente sarà in grado di conoscere:

- Sprite, sfondi Stage e costumi
- Utilizzare i blocchi istruzioni e comandi dell'interfaccia di Scratch
- Edit di Scratch e modificare le immagini da utilizzare come sprite
- Utilizzo dei suoni
- Utilizzo di più sprite conteporaneamente

Competenze, abilità

Al termine dell'Unità Didattica l'alunno dovrà essere in grado di:

- Riconoscere le caratteristiche fondamentali delle istruzioni che possono comporre un algoritmo e le strategie risolutive sviluppando nel linguaggio di programmazione Scratch
- caricare e creare sprite e sfondi
- Risolvere un problema utilizzando i giusti blocchi per ottenere i risultati desiderati su sprite, costumi e sfondi.
- cambiare scenari
- riproporre la stessa soluzione su un problema simile.

2.4 Strumenti, tempi, attività didattiche

Srumenti, sussidi didattici

- Libri di testo - dispense
- Presentazioni multimediali
- Lavagna luminosa (LIM)
- Software Scratch 1.4 o 2.0

Tempi: 3 SETTIMANE : 18 ore

LEZIONE FRONTALE 6 ore

ESERCITAZIONI 6 ore

VERIFICHE: 6 ore (1h test + 2verifica lab + 2h recupero +1h correzione)

2.5 Piano di lavoro

fase	articolazione	strategie e contenuti	sussidi didattici	n.ore
1	accertamento prerequisiti	test di ingresso	materiale didattico	1
2	Attività frontale (F)-Laboratorio(L)	F: primi elementi di programmazione,problemi e algoritmi (2h-F) F: l'interfaccia di Scratch, sprite, sfondi e costumi (2h-F) L: movimentodel gattino e il cambio colore e costume (2h-L) F:variabili, sequenza, selezione e cicli (2h-F) L: movimenti e suoni: il breaker(2-L) L:Stage, sfondi e più sprite:l'inseguimento (2h-L) L:più sprite che iteragiscono:l'acquario virtuale (2h-L)	Lim,materiale didattico, Software Scratch 1.4	12
3	Verifica lab	esercizi per progressiva difficoltà a scelta 1-labirinto 2-gare 3-pianoforte	Griglia voti-Tayler es.1: 6-7 Es 2:7-8 Es 3:8-10	2
4	Recupero	Esercizio Ping Pong	Griglia Tyler	2
5	Correzione Verifica	----------	Griglia Tyler	1

2.6 Griglia di Tyler

		Obiettivi					
		La differenza tra Sprite, Costume e Sfondo sullo Stage	Impara a usare gli Sprite per i movimenti e cambiare il	Impara a cambiare scenario e quindi a intercambiare lo sfondo sullo Stage e/o gli	Riconoscere e distinguere tra i diversi blocchi quelli che più si adattano alle esigenze del progetto	Caricare o creare sprite, sfondi e costumi	
Contenuti							
	Sfondo	x		x		x	
	Sprite	x	x			x	
	Costumi	x	x			x	
	istruzioni		x	x	x		

2.7 Esercizi-laboratorio

2.7.1 Esercizio 1 Come ti chiami?

Creare un fumetto in cui il gattino interagisce chiedendo il tuo nome

2.7.2 Esercizio 2 Come cresco!

Creare uno sprite dove il gattino aumenta di dimensione e varia colore progressivamente.

2.7.3 Esercizio 3 Il breaker

Realizzare un breaker che si muove da destra verso sinistra emettendo un suono di tamburo con uno sfondo da discoteca

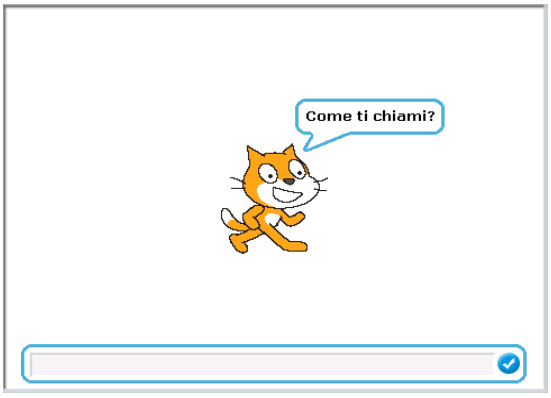
2.7.4 Esercizio 4 L'inseguimento marino

Realizzare uno sfondo marino e due sprite che interagiscono di un pescecane che insegue un'altro pesce

2.7.5 Esercizio 5 L'acquario virtuale

Realizzare su uno sfondo marino più sprite che nuotano da una parte all'altra dell'acquario senza fermarsi

2.7.1 Esercizio 1: Come ti chiami? Creare un fumetto in cui il gattino interagisce chiedendo il tuo nome



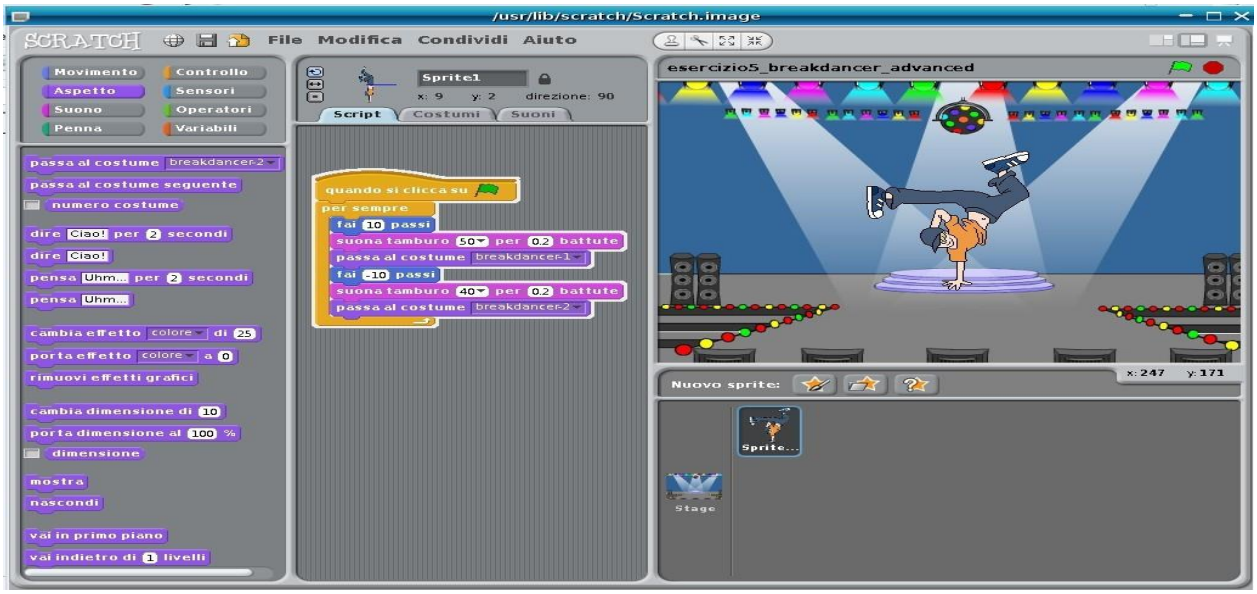
In questo semplice esercizio di inizio programmazione si utilizza il blocco viola "**dire..**" che serve per visualizzare la scritta nello schermo"

2.7.2 Esercizio 2- Come cresco!: sviluppare un programma che consente allo sprite del gattino di crescere progressivamente e di cambiare colore.

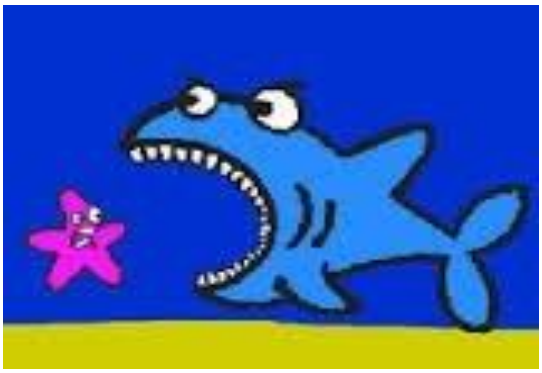
In questo esercizio utlizziamo un ciclo ripetitivo " **per sempre**" per far cambiare in continuazione colore e dimensione al gattino. All'interno del ciclo facciamo eseguire l'istruzione di cambiare effetto colore e dimensione di un valore che gli attribuiamo, ad esempio 50 per il colore e 30 per la dimensione. Il risultato è il seguente:

2.7.3 Esercizio 3 Il breaker.

In questo esercizio utilizziamo un ciclo ripeti per sempre alcune istruzioni come far muovere il breaker, far suonare il tamburo ad ogni movimento, modificare il costume e passare al movimento successivo (il breaker va avanti (10) e indietro(-10)



Esercizio 4: l'inseguimento



Proviamo ora ad usare due sprite conteporaneamente, presi dalla libreria di Scratch, sotto lo Stage e sviluppiamo il programma di uno squalo che insegue un'altro pesce.

2.7.4 Esercizio 4: l'inseguimento

In quest'esercizio utilizziamo due sprite. Prima si inseriscono gli sprite relativi alla creature acquatiche che intendiamo far muovere e i relativi script con i quali programmiamo il movimento.Eseguiamo dei cicli iterattivi "per sempre" per far muovere gli sprite di quanto desideriamo, e fargli rimbalzare con l'apposito blocco quando si tocca il bordo. Programmiamo gli sprite per farli girare appunto quando toccano il fondo altrimenti si visualizzerebbe l'immagine al contrario. Nell'esercio successivo, dell'acquario virtuale spiego passo per passo cosa fare.

2.7.5 Esercizio 5 L'acquario virtuale

Esercizio livello 3 Sprite, Sfondi e Costumi

Nell' esercizio successivo creiamo un' acquario virtuale.

I passi da seguire sono i seguenti:

1- Iniziamo a cancellare dallo Stage lo sprite del gatto dopo aver selezionato le forbici tra i pulsanti sopra lo Stage.

2- Aprire la libreria degli Sprite dalla quale selezionare la categoria Animali

3- Fare click sullo sprite a forma di pesce che preferisci e ripeti l'operazione più volte, inserendo ogni volta nello Stage un nuovo pesciolino.

4-Ridimensiona gli sprite selezionando gli appositi pulsanti collocati al

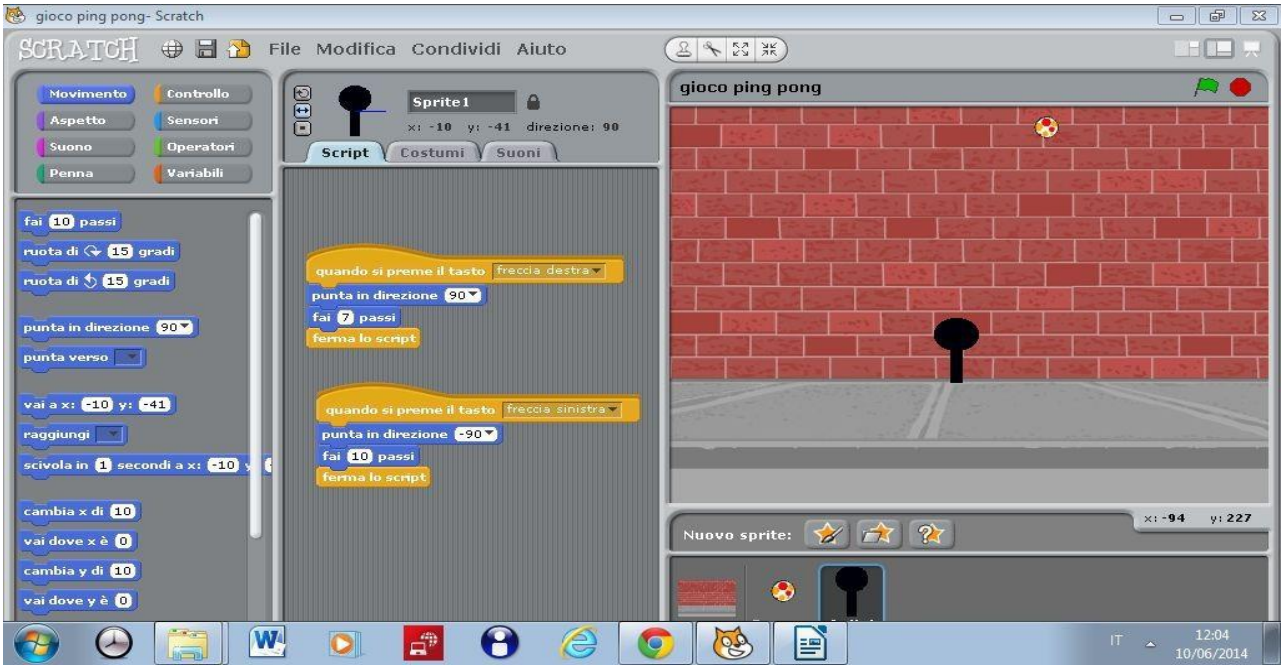
centro della cornice di scratch in alto.
5- Ora scegli uno sfondo, selezionando la categoria sfondi e scegli quello denominato Underwater (se vuoi puoi disegnare il tuo sfondo usando l'editor oppure importarlo attraverso la webcam del computer
6- A questo punto seleziona uno sprite dei pesci per volta e associa lo script corrispondente; prova ad associare a ciascuno dei tre pesciolini uno dei seguenti codici, inserendolo nell'area script di ciascuno di essi.
7- Attraverso il blocco **per sempre** della categoria **Controllo** lo sprite ripete senza sosta lo spostamento di 1,2 o 3 passi, e mediante il blocco **fai...passi** della categoria **movimento**, sino ad arrivare al bordo dello stage; fai poi rimbalzare lo sprite attraverso il blocco **rimbalza quando tocchi il bordo**, anch'esso della categoria movimento.
8- Per far si che i pesci, toccato il bordo dell'acquario, si voltino per tornare indietro, seleziona uno sprite per volta e imposta lo **stile rotazione** facendo click sul pulsante **voltati solo a sinistra-destra** e in questo modo gli sprite si muoveranno andando da sinistra verso destra e viceversa come se fossero tanti pesciolini in un acquario
9- Ecco il risultato che ottieni facendo click sulla bandierina verde

2.8 Verifica Recupero Il Ping Pong

Creare il Gioco del Ping Pong, dove il gioco ha inizio con la pressione della barra spaziatrice mentre la racchetta si sposterà da destra verso sinistra attraverso la pressione dei tasti della tastiera.
Devi colpire la pallina e la devi far rimbalzare contro il muro senza farle toccare il marciapiedi altrimenti...
Sviluppiamo il progetto passo a passo:
1- Cancelliamo dallo Stage il gattino e apriamo la libreria degli sprite dalla quale selezioniamo la categoria cose e lo sprite Tennis Ball. (o la pallina che vuoi)
2- Disegna lo sprite della racchetta attraverso l'editor del disegno.
3- Scegli uno sfondo dalla libreria in Esrerni e lo sfondo brickwall1.
4-Posiziona lo sprite della racchetta appena sopra il marciapiede.
5- Seleziona le informazioni associate allo Sprite e imposta lo stile di rotazione col pulsante voltati solo a sinistra-destra
6- aggiungi allo sprite della racchetta gli script
(immagine 1 e 2 di Ping Pong)

Simulazione Gioco Ping Pong

Es 4- Gioco del ping pong a muro



2.9 VERIFICHE - VALUTAZIONE

Le verifiche di valutazione (6 ore) sono così suddivise:
Verifica Prerequisiti:test V/F (1 ORA)
Verifica laboratorio e verifica sommativa: Esercizi sulla risoluzione di semplici algoritmi con utilizzo di Sprite, sfondi e costumi e relative modifiche richieste (2 ORE)
Verifica Recupero: Esercizio (2 ORA)
Correzione Verifica (1 ORA)
Ausilio di griglia di valutazione

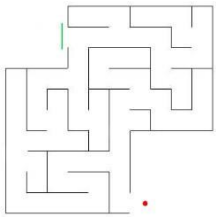
2.9.1 Verifica Prerequisiti

Test sui prerequisiti (test a risposta unica)

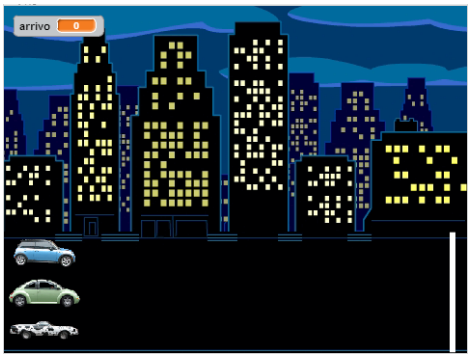
- Come avviene la scrittura di un'istruzione?
    1. Trascinando un blocco dall'area dei comandi
    2. Digitando le istruzioni
    3. Assegnano allo stage le istruzioni digitate
- Dove si trovano i blocchi delle istruzioni
    1. Cliccando nel TAB script
    2. Facendo doppio click sullo sprite
    3. Facendo doppio click sullo sullo stage
- Che tipi di blocchi contiene la sezione "Controllo"
    1. Contiene blocchi di cicli, controlli
    2. Contiene blocchi di istruzioni su movimenti e controlli
    3. Contiene blocchi di istruzioni sui controlli da tastiera

2.9.2 Verifica labirinti

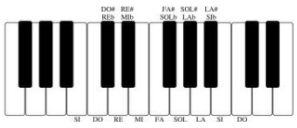
- **Cerchiamo su internet 3 immagini di labirinti e carichiamoli come sfondo. Creiamo uno sprite a forma di pallina e assegnamo allo sprite i movimenti eseguibili mediante le frecce da tastiera che gli consentano di muoversi nel labirinto e trovare l'uscita. Ogni volta che urta la parete del labirinti (colore nero) lo sprite rimbalza. Arrivato alla fine del labirinto il gioco ricomincia e viene caricato un altro labirinto.**



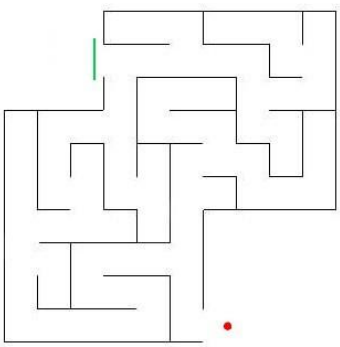
- **2.9.3 Gare automobilistiche: Creiamo uno scenario di una gara automobilistica. Caricare 3 sprite che rappresentano automobili.**

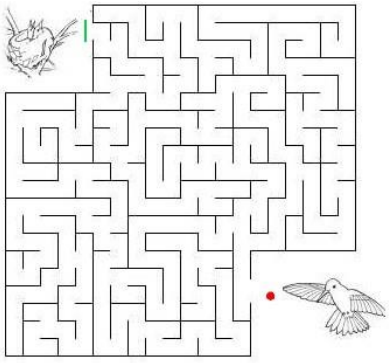


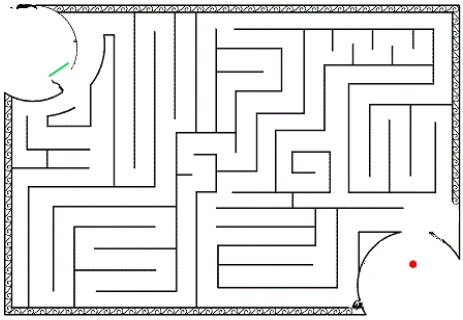
Posizioniamoli da un lato dello schermo e dall'altro creiamo una linea di arrivo mediante un altro sprite. Aquesto punto facciamoli gareggiare generando un numero casuale tra 0 e 10 da assegnare al movimento di ciascuno sprite. Il primo sprite che tocca la linea di arrivo vince.

- **2.9.4 Carichiamo una immagine di sfondo che rappresenta una tastiera**  **da pianoforte. Carichiamo uno sprite che**   **rappresenta un dito e che ha con 2 costumi: dito diritto , dito giù . Sapendo che nella notazione americana il DO=C**  **RE=D MI=E FA=F SOL=G LA=A SI=B assegnare allo sprite mano lo script che indica che alla pressione del tasto C sulla tastiera del PC la mano si deve posizionare in corrispondenza del tasto DO, deve effettuare una cambio costume per simulare la pressione del tasto sulla tastiera del pianoforte e deve emettere il relativo suono di DO (C nella notazione americana) usando lo script . Ripetere questo script per tutte le altre note**

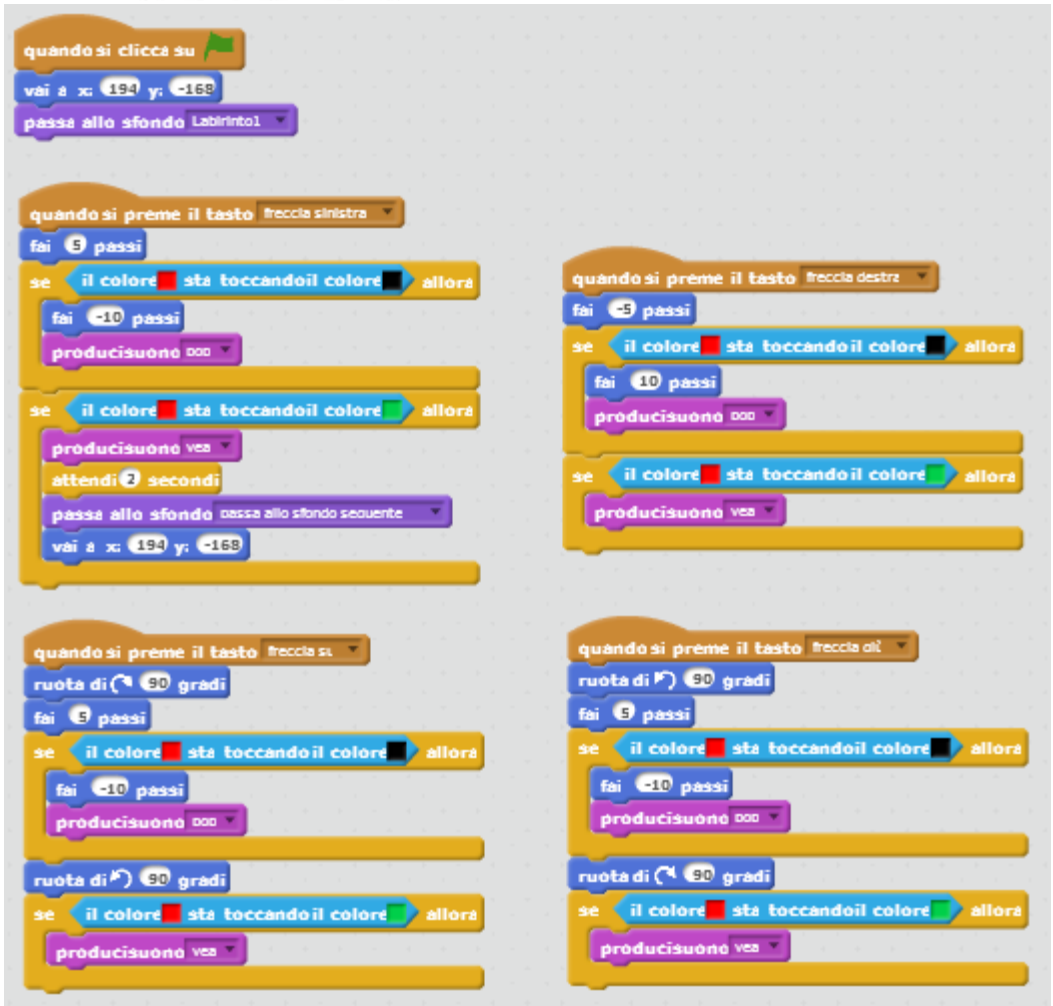
Possibile soluzione per labirinto



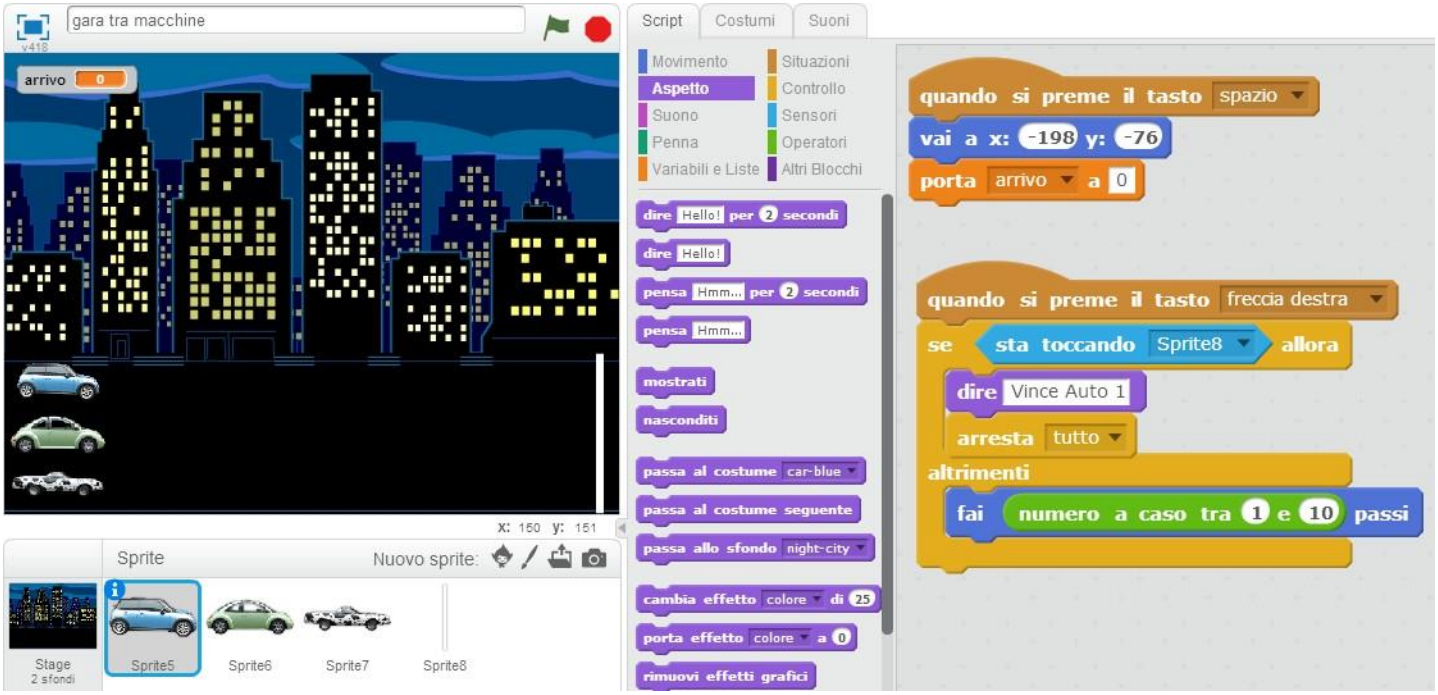




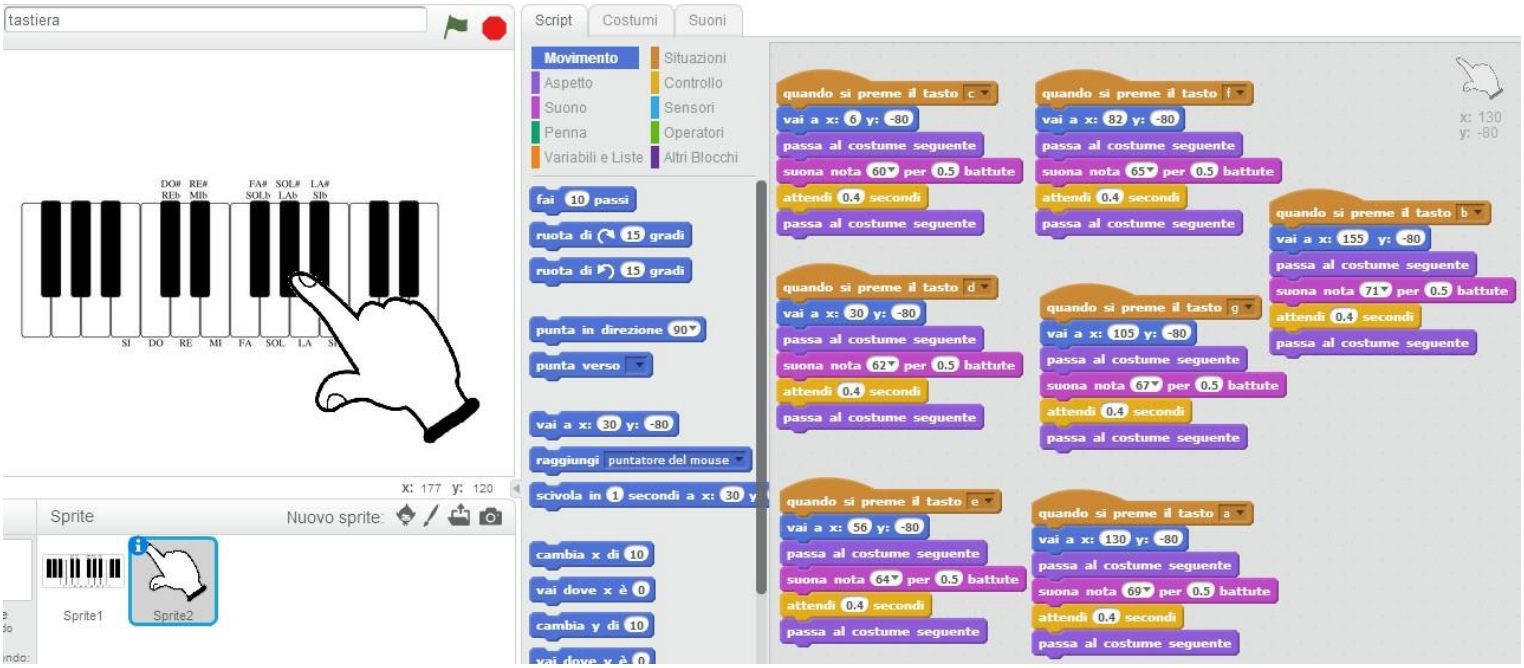

Sprite
Stage
4 sfondi
Sprite1
Nuovo sfondo:


quando si clicca su
vai a x: 194 y: -168
passa allo sfondo Labirinto1
quando si preme il tasto freccia sinistra
fai 5 passi
se il colore sta toccando il colore allora
fai -10 passi
produci suono doo
se il colore sta toccando il colore allora
produci suono vea
attendi 2 secondi
passa allo sfondo passa allo sfondo seguente
vai a x: 194 y: -168
quando si preme il tasto freccia destra
fai -5 passi
se il colore sta toccando il colore allora
fai 10 passi
produci suono doo
se il colore sta toccando il colore allora
produci suono vea
quando si preme il tasto freccia su
ruota di 90 gradi
fai 5 passi
se il colore sta toccando il colore allora
fai -10 passi
produci suono doo
ruota di 90 gradi
se il colore sta toccando il colore allora
produci suono vea
quando si preme il tasto freccia giù
ruota di 90 gradi
fai 5 passi
se il colore sta toccando il colore allora
fai -10 passi
produci suono doo
ruota di 90 gradi
se il colore sta toccando il colore allora
produci suono vea

2.9.4 Possibile soluzione per gara auto



Possibile soluzione per Pianoforte

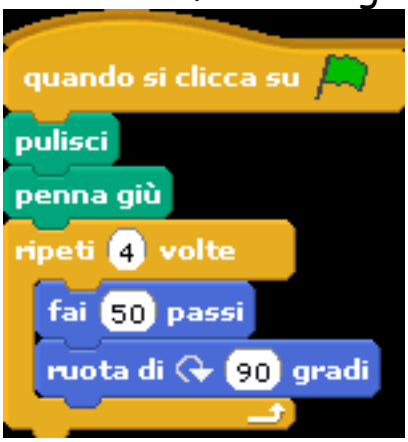


Compiti per casa

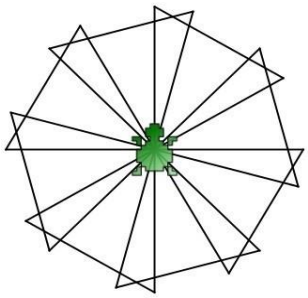
Ai fini di poter approfondire i concetti introdotti nella programmazione di Scratch si consigliano i seguenti compiti per casa:

1.Eseguire i seguenti esercizi per progressiva difficoltà

Esercizio 1: In questo Esercizio si chiede di utilizzare lo sprite del gattino e di fare disegnare un quadrato



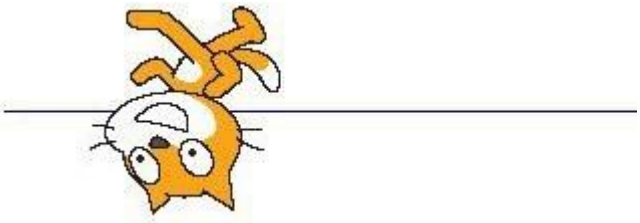
Esercizio 2- In quest'esercizio, si chiede di disegnare un fiore con dei triangoli equilateri



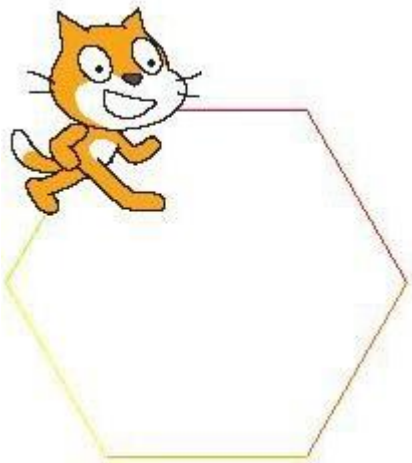
Esercizio 3 - Disegnare una spirale quadrata con Scratch

Esercizio 4 - Disegnare un poligono con Scratch

Esercizio 5 – far camminare il gattino all'infinito



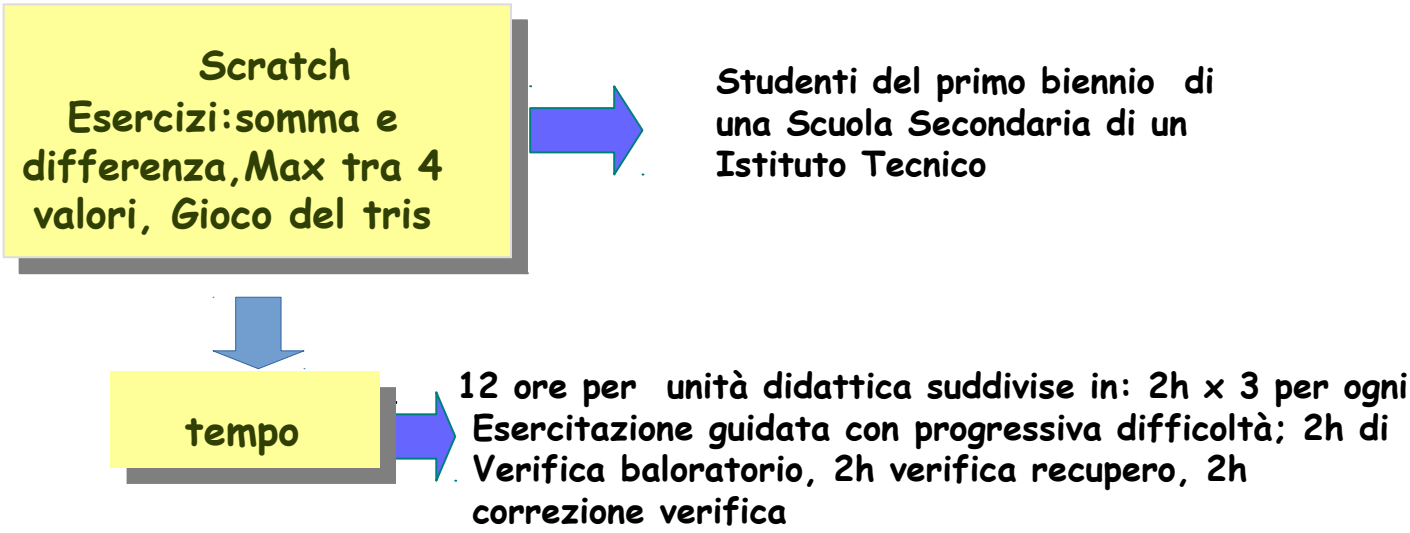
Esercizio 6 Esagonono con colori diversi



Capitolo 3 Primi elementi di programmazione

3.1 Il contesto

L'Unità Didattica è rivolta a studenti di un Istituto Tecnico Superiore che ha inserito nel proprio curricolo di studio Informatica di base, o affine.
L'unità Didattica è articolata in ore ….. da suddividere in …..
Gli esercizi che vengono affrontati sono esercizi di base della programmazione, spesso affrontati dai ragazzi con un certo livello di difficoltà qualora vengano sviluppati in linguaggi quali ad esempio il C, o C++, mentre con Scratch l'implementazione risulta più semplice e intuitiva



3.2 Prerequisiti

Per affrontare questa Unità Didattica, in generale è importante avere acquisito le conoscenze di base di:
- sistemi operativi
- hardware e software
- modello e sistema
- sviluppo algoritmico
-diagrammi di flusso
-psudocodifica
-i dati, le costanti
-le variabili
-sequenza, selezione e ripetizione
-Le strutture di controllo e i cicli

3.3 Conoscenze, competenze, abilità

- Le conoscenze

Al termine dell'Unità Didattica gli alunni dovrebbero acquisire le conoscenze di Primi Elementi di Programmazione, saper formalizzare un problema e adottare una efficiente strategia risolutiva. Affrontato questo primo fondamentalepasso dovrebbero essere in grado di fare un minimo di programmazione, e quindi saper utilizzare le variabili, le sequenze,la selezione, i cicli ripetitivi.

- Competenze e abilità

Gli alunni devono essere in grado di:

- Riconoscere le caratteristiche fondamentali delle istruzioni che possono comporre un algoritmo e le strategie risolutive
- Risolvere un problema e riuscire a superare l'ostacolo "programmazione passando dall'algoritmo strutturato al programma
- Riconoscere, individuare le strutture di confronto, gli operatori logici e booleani, le strutture ripetitive

3.4 Strumenti, attività didattiche

ATTIVITÀ DIDATTICHE ⇨

*Lezioni frontali in classe
Esercitazioni pratiche
Esercitazione di gruppo

SUSSIDI DIDATTICI ⇨

Libri di testo - dispense
Presentazioni multimediali
Lavagna luminosa (LIM)
Software Diagram Designer o Draw.Io
Software Scratch 1.4 o 2.0

3.5 Piano di lavoro

FASI Attivita' didattiche	CONTENUTI- Esercitazioni laboratorio	SUPPORTI- Sussidi didattici	N_ORE	PUNTEGGIO Griglia valutazione
Esercizio 1	somma e diff	D.D. Scratch	2	sufficiente discreto
Esercizio 2	Max e Min tra 4 valori	D.D. Scratch	2	buono
Esercizio 3	Gioco del tris	D.D. Scratch	2	distinto ottimo
Verifiche:livello 1 Livello 2 Livello 3	Somma Pari I triangoli	Scratch	2	griglia voti
Recupero	Il max tra 3 valori	Scratch	2	griglia voti
correzione	Verifica	Lim, D.D.Scratch	2	griglia
Totale Argomento V=verifica L=laboratorio R=recupero CV=correzione verifica	primi elementi di programmazione	Lim, Diagram D. Scratch	12 (6L 2V 2 R 2CV)	griglia

3.6 Griglia di Tyler

		Obiettivi					
		Risoluzione di un problema in maniera efficiente	Inserire le istruzioni n sequenza e fare eassegnazioni	Saper effettuare dei confronti con gli operatori opportuni	Usare le istruzioni di Selezione Se e Altrimenti	Saper utilizzare le variabili	Distinguere tra dati in input e risultati in output
Contenuti							
	Problema	x					
	Sequenza	x	x	x	x	x	x
	Selezione	x		x	x	x	
	Istruzioni	x	x	x	x	x	x

3.7 Attività proposte

3.7.1 Esercizio 1, Somma e differenza: dati due valori A e B, se A è maggiore di B calcolare la somma altrimenti calcolare la differenza

3.7.2 Esercizio 2 il maggiore tra 4. Dati 4 valori in ingresso determinare il minore e il maggiore.

3.7.3 Esercizio 3, il gioco del tris. Sviluppa un programmino che realizzi il gioco del tris

Punteggio: (risolti sia l'algoritmo che il programma)

1 Esercizio voto da 6 a 7

2 esercizio Voto da 7 a 8

3 Esercizio Voto da 8 a 10

Risoluzioni

3.7.1 Esercizio 1, Somma: dati due valori A e B, se A è maggiore di B calcolare la somma altrimenti calcolare la differenza

Analisi

Il problema ci pone il confronto tra due valori A e B.

Per fare il confronto tra i due valori usare la struttura tra i **controlli Se e Altrimenti.**

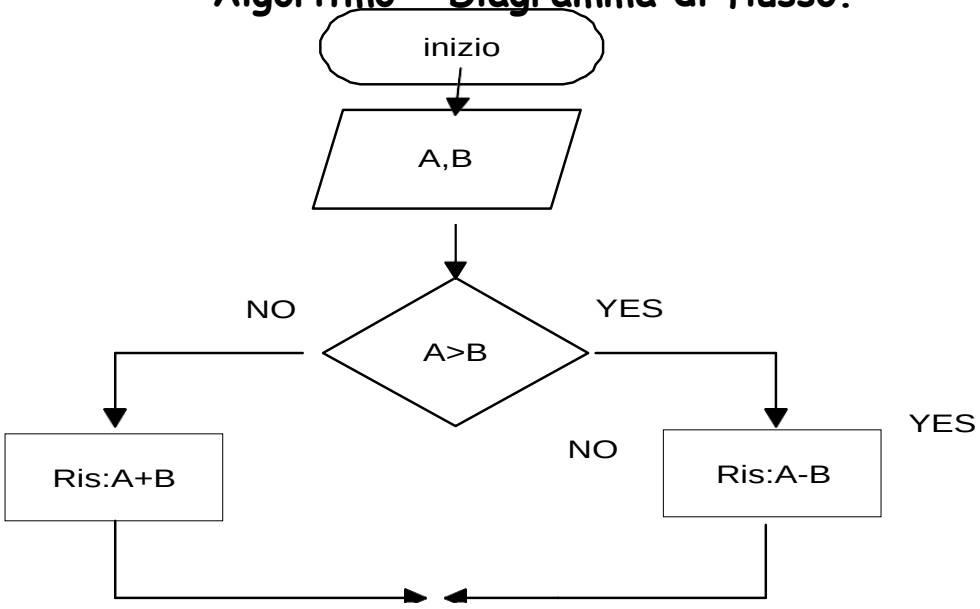
Tabella di traccia

input A , B

output il più grande dei 2

relazione I/O Se A>B allora fare A-B
altrimenti Se B>A allora fare A+B

Algoritmo – Diagramma di flusso.



Sviluppo Software: dati 2 valori, se A>B fare la differenza altrimenti fare la somma

3.7.2 Esercizio 2, Il Massimo tra 4 valori: dati quattro valori: A,B,C,D calcolare il massimo e il minimo

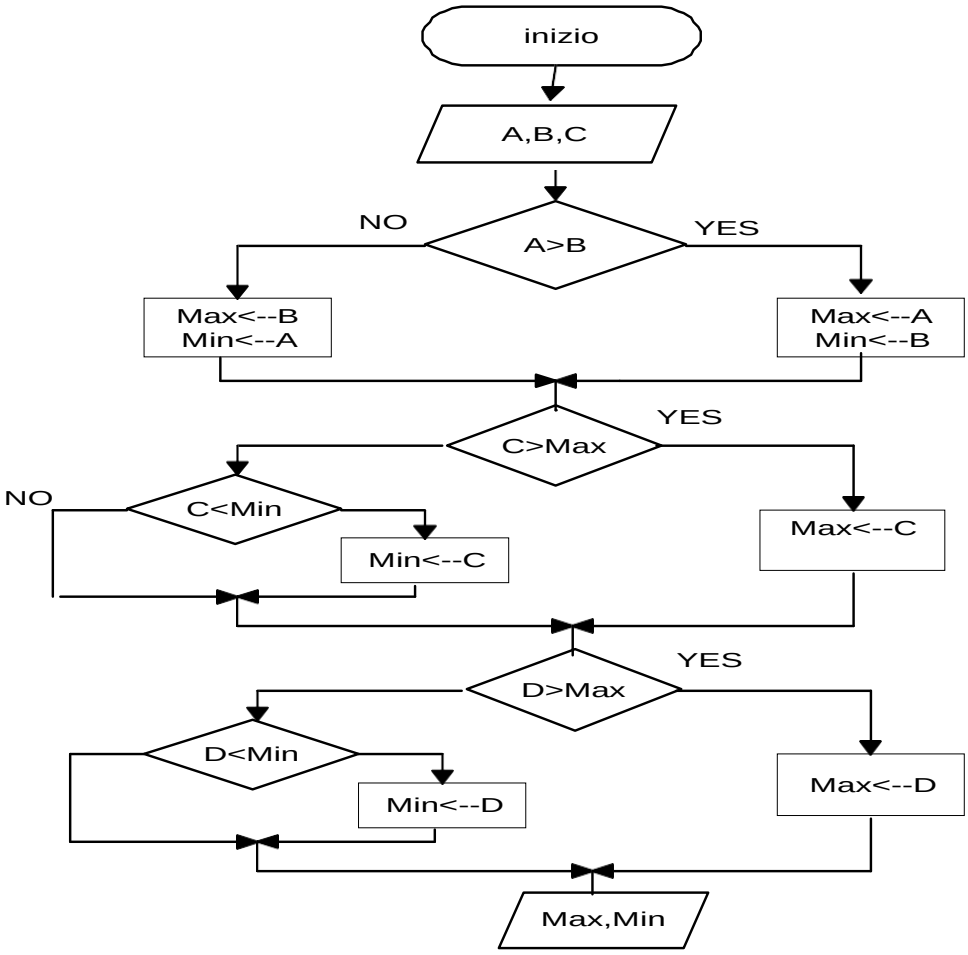
Analisi

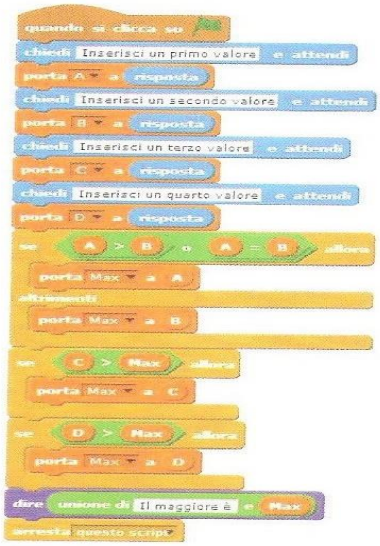
Il problema ci pone il confronto tra quattro valori A, B,C,D
Occorre usare 2 variabili ausiliarie Max e Min e fare gli opportuni controlli.

Tabella di traccia

input A,B,C, D
output Max e Min
relazione I/O Se A>B Max prende A e Min prende B, altrimenti Max prende B e Min prende A Se C>Max Max prende C, altrimenti controllo C<Min, se vero Min prende C Se D>MAx Max prende D, altrimenti controllo se D<Min, se vero Min prende D.
risultato: Max e Min

Algoritmo – Diagramma di flusso: Max e Min tra 4 valori





Sviluppo Software:dati 4 valori determinare Max e Min

3.7.3 Esercizio 3, il gioco del tris. Sviluppa un programmino che realizzi il gioco del tris.

Il gioco del tris è un gioco in cui ci sono 2 giocatori, e dove vince chi totalizza tre simboli uguali xxx.

I passi da seguire sono i seguenti:

1- Selezionare l'oggetto Stage e poi la scheda Sfondi e scegli la categoria e lo sfondo che più ti piace.

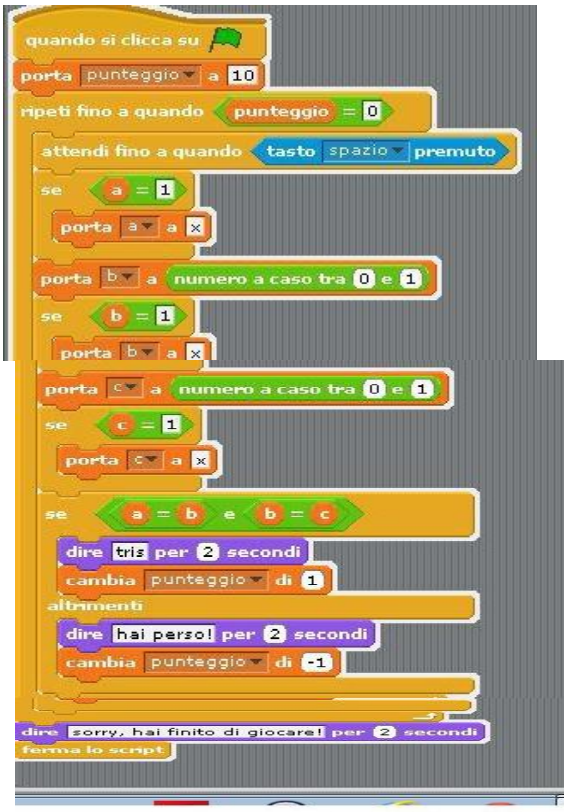
2- Si creano quattro variabili: A,B,C, per contenere i tre numeri casuali e la variabile Punteggio per contenere il punteggio assegnato in partenza (supposto uguale a 10) e quello residuo, e lascia attiva la spunta accanto al loro identificatore in modo che ne possa visualizzare il valore contenuto

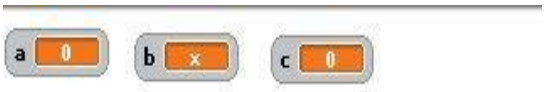
3- Il codice: Si assegna alla variabile punteggio un valore iniziale pari a 10 e il programma attende che il giocatore prema la barra spaziatrice per generale 3 numeri a caso (tra 0 e 1) e assegnarli alle variabili A,B,C. A questo punto per ciascuna delle tre variabili viene utilizzato un blocco se...allora per potre visualizzare il simbolo x, associato al valore 1, nell'area gioco. Solo se hai fatto tris (A=B e B=C) il punteggio viene aumentato di 1 punto altrimenti viene decrementato.

Il programma termina quando darà esito positivo il controllo all'interno del ciclo (punteggio=0) e in questo ultimo caso verrà fermato il gioco. Il risulato ottenuto è mostrato successivamente:

I

blocchi del progetto

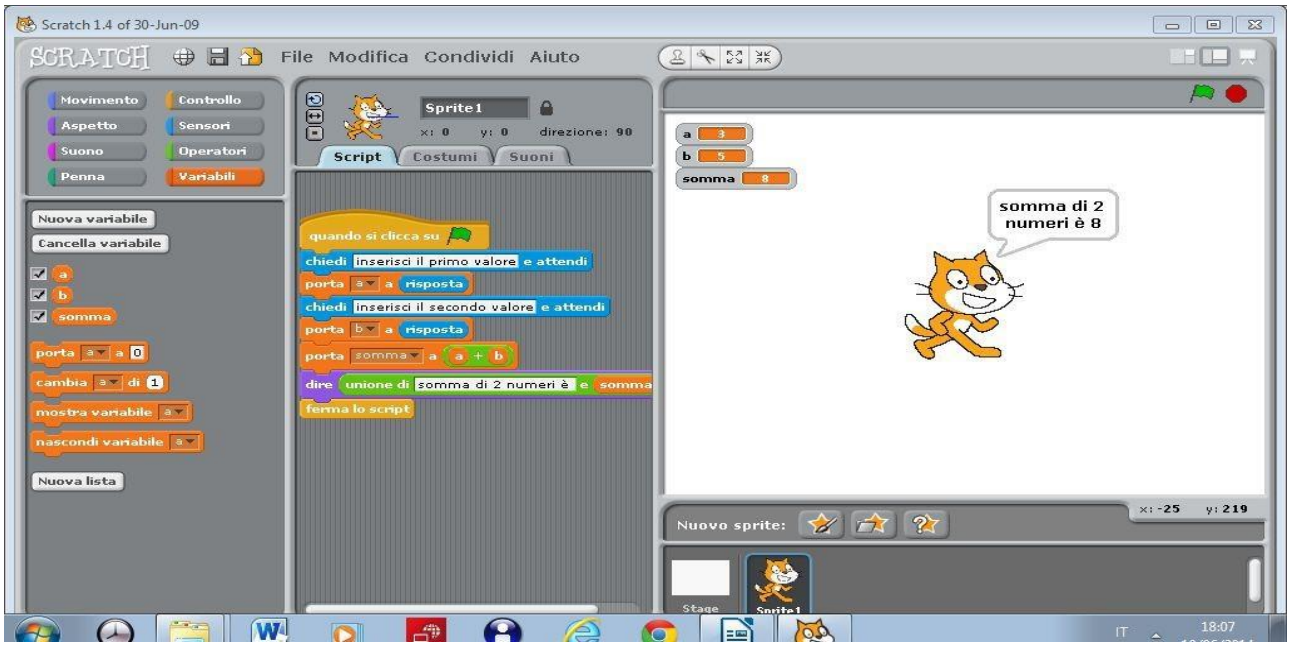




3.8 Verifiche, Valutazione

Verifica laboratorio e verifica sommativa: Esercizi sulla risoluzione di semplici problemi con strutture ripetitive. Analizzare il problema e sviluppare il programma con Scratch (2 ORE).
Correzione Verifica (1 ORA).
Ausilio: Griglia di Tyler.
Ora fai tu: Si propongono ora 3 tipi di Verifica per progressiva difficoltà, la cui valutazione è stabilita secondo un criterio dettato dalla tipologia di difficoltà dell'esercizio:
Verifica 1, **Somma**, difficoltà bassa: **dati in ingresso 2 valori fare la somma**
Verifica 2,Pari, difficoltà media: **visualizzare tutti i numeri pari da 0 a 20**
Verifica 3, **I tre triangoli**, difficoltà maggiore: date le lunghezze di tre lati visualizzare il tipo di triangolo, isoscele, equilatero o scaleno.

3.8.1 Verifica 1 Somma: dati in ingresso 2 valori visualizzare la somma con messaggio dello sprite



In questo problema i passi da seguire sono:
Chiedere l'inserimento dei 2 valori e calcolare la somma con gli operatori matematici e infine visualizzare il messaggio "la somma dei 2 valori è.." col blocco Unione.

3.8.2 Verifica 2: visualizzare tutti i numeri pari da 0 a 20

In questo problema i passi da seguire sono:

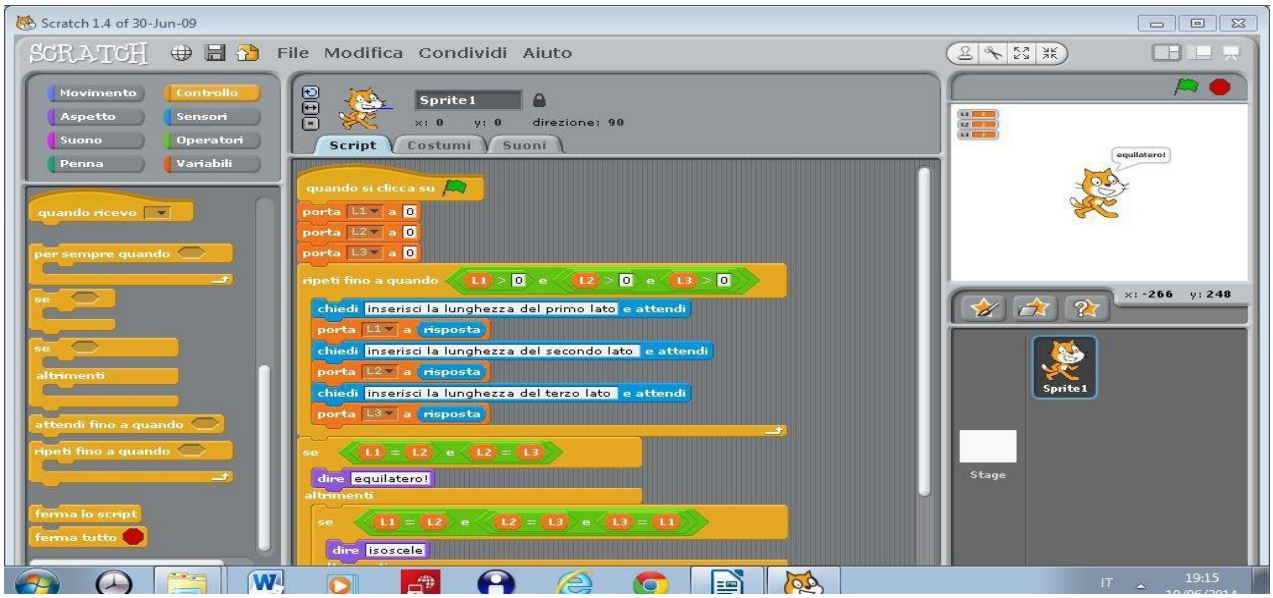
In questo problema i passi da seguire sono:

1-inizializzare una variabile N a zero che incrementa di 2 in 2 all'interno di un ciclo eseguito logicamente per 10 volte (dobbiamo arrivare a 20)

2- calcolare se il numero è pari, e in caso affermativo visualizzarlo, altrimenti il valore è dispari e non viene considerato.

3- il ciclo si ferma quando N arriva a 20.

3.8.3 Verifica n.3 maggiore difficoltà. Ricevute in ingresso le lunghezze dei tre lati di un triangolo, dire se si tratta di un triangolo equilatero,isoscele o scaleno



Nel problema dei 3 tipi di triangoli i passi da seguire sono:
1- inserire le lunghezze dei tre lati
2- fare tutti i controlli
3- eseguire un ciclo che ripete tutte le operazioni fino a quando le misure dei lati sono tutte e tre conteporaneamente meggiori di 0, che è la condizione affinché si tratti di un trinagolo
4- eseguire i controlli che soddisfano le condizioni secondo le quali si hanno i tre tipi di triangoli.

3.9 Verifica di Recupero

Inserire tre valori e calcolare il maggiore tra i tre. Sviluppare analisi, algoritmo con diagramma di flusso e tabella di traccia e programma.

Analisi

Quest'esercizio di difficoltà 2 propone dati 3 valori numerici determinare il maggiore tra i 3. Sviluppare in tre punti fondamentali: 1_analisi, 2_diagramma di flusso, 3_software

1:_Analisi del problema: date 3 variabili, A, B, C, determinare il più grande tra i 3 valori. Per sviluppare il nostro ragionamento individuamo una tabella di traccia che mostra una traccia delle istruzioni che dovremmo eseguire

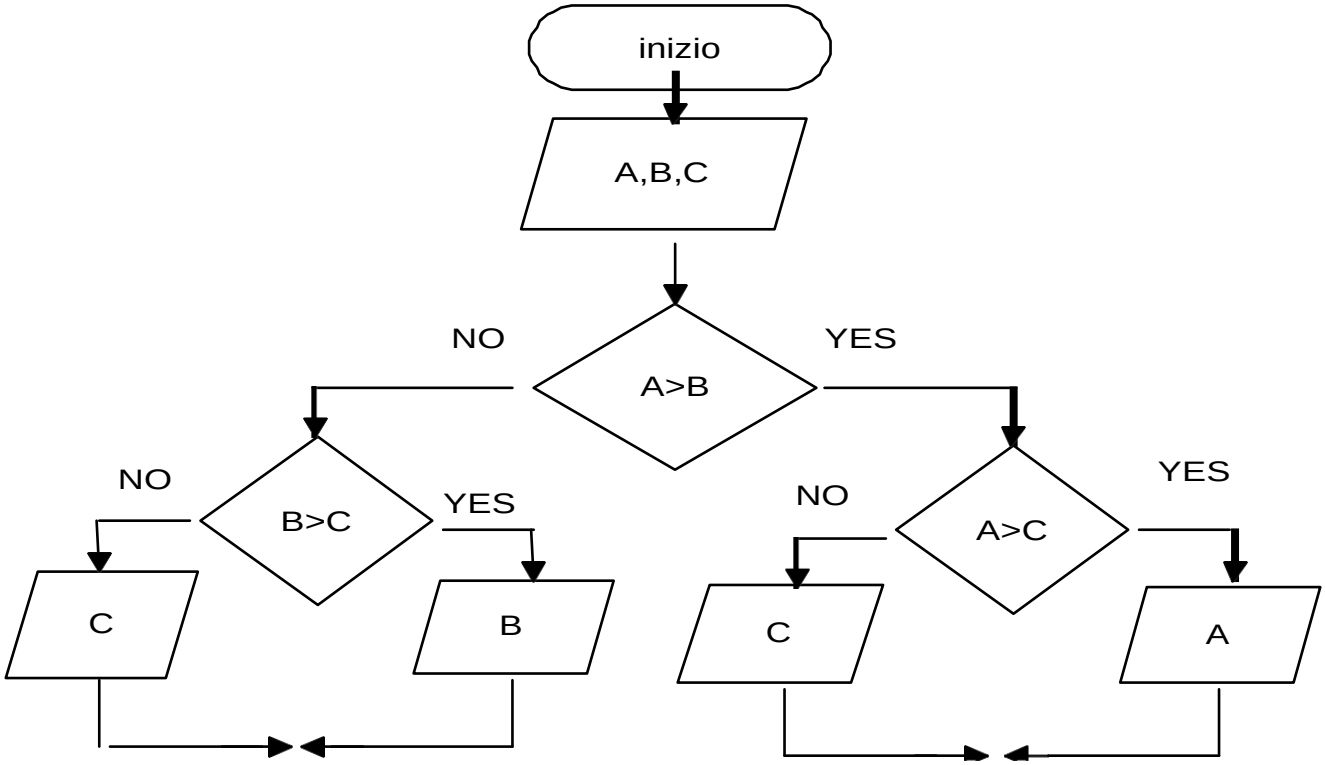
Tabella di traccia

input A B C

output il più grande dei 3

relazione I/O Se A>B e A>C allora il più grande è A
Se B>A e B>C allora il più grande è B
Se C> A e C>B allora il più grande è C

Diagramma di flusso, il Maggiore tra 3 valori



Sviluppo del programma

Conclusioni

Gli esercizi proposti in laboratorio con relativi algoritmi risolutivi, diagramma di flusso e tabella di traccia costituiscono una prima base di programmazione, fatta di Analisi, fondamentale per capire un problema, diagramma di flusso, determinante soprattutto in questa prima fase di approccio alla programmazione, per sviluppare il software e con il supporto della tabella di traccia che serve a costruire le fasi del nostro algoritmo distinte in Input, Output, Risultati.

Le tre verifiche proposte mostrano agli alunni quali saranno i voti che prenderanno se risolveranno correttamente il problema con l'algoritmo risolutivo.

I contenuti sono stati affrontati nell'Unità didattica precedente e poi approfonditi nelle esercitazioni.

Gli approcci risolutivi di un problema possono essere diversi ma ciò che è importante è rispettare ciò che ci viene insegnato arrivando a una delle possibili soluzioni utilizzando alcuni tra gli strumenti proposti.

Ricordiamoci sempre di:

- analizzare bene il problema;
- scomporlo in sottoproblemi;
- costruire una descrizione delle azioni distinguendo tra azioni input e risultati in output (tabella descrizione variabili);
- eseguire un diagramma di flusso con un software a disposizione Free come Draw o Diagram Designer;
- fare una tabella di traccia che ci aiuta a capire l'andamento delle azioni;
- sviluppare il software riconoscendo e seguendo i costrutti fondamentali dell'interfaccia di Scratch.

FINE U.D. Primi elementi di programmazione

Capitolo 4 "Primi elementi di programmazione" i contenuti

4.1 Dal problema al programma"

Programma: insieme finito di istruzioni, che eseguite in sequenza permettono di elaborare i dati in ingresso per ottenere in uscita risultati richiesti dall'elaborazione in modo da risolvere un determinato problema

Formalizzare un problema

Per passare dal problema da risolvere al programma che lo risolve automaticamente occorre quindi formalizzare il problema, che inizialmente in linguaggio naturale deve passare attraverso varie fasi di trasformazione volte ad ottenere il programma che risolve il problema di partenza.

Analizzare un problema significa individuare dopo un'attenta lettura dei dati in ingresso, ossia le informazioni fornite in ingresso e i vincoli di integrità, ossia le condizioni che gli input devono rispettare per essere accettati dal programma.

Esempio di un videogioco, il labirinto, vogliamo spostarci da destra verso sinistra con la tastiera; non potremmo che utilizzare le frecce.

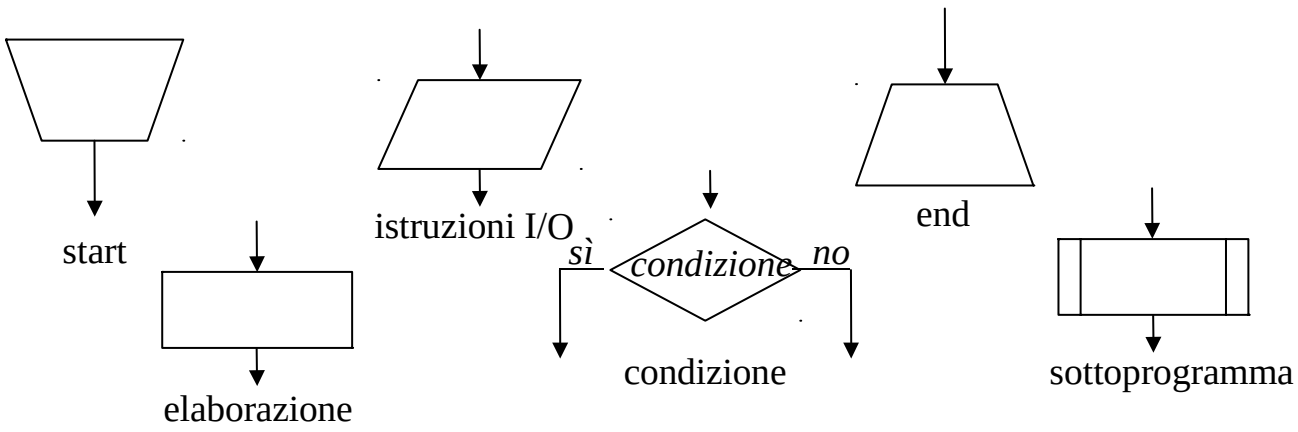
4.2 Sviluppo dell'algoritmo

Dopo aver individuato le relazioni tra Input e Output, occorre individuare un metodo ottimale di risoluzione del problema di partenza cioè la sequenza di azioni, che seguendo le indicazioni date dalla relazione I/O permetta di trasformare i i dati in ingresso in risultati,cioè in dati in uscita.

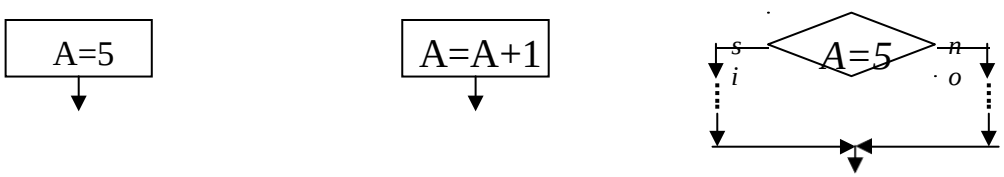
Ma l'elaboratore non è in grado di trovare il metodo risolutore sta al **programmatore** il compito di "indicare" le azioni da eseguire e le regole da seguire per ottenere il risultato voluto, cioè l'algoritmo risolutore del problema.
Un algoritmo è una descrizione di un insieme finito di passi, istruzioni che devono essere eseguite per risolvere un dato problema e per raggiungere un risultato definito.
Ma gli Algoritmi per poter essere poi sviluppati in un linguaggio di programmazione possono avvalersi di le istruzioni rappresentate mediante uno schema di flusso (o diagramma a blocchi) che è la rappresentazione grafica di un algoritmo realizzata mediante l'utilizzo di simboli, la cui forma dipende dal tipo di azione che si vuole descrivere, uniti da frecce che rappresentano il flusso dell'esecuzione delle istruzioni che compongono l'algoritmo.

4.3 I Diagrammi a blocchi

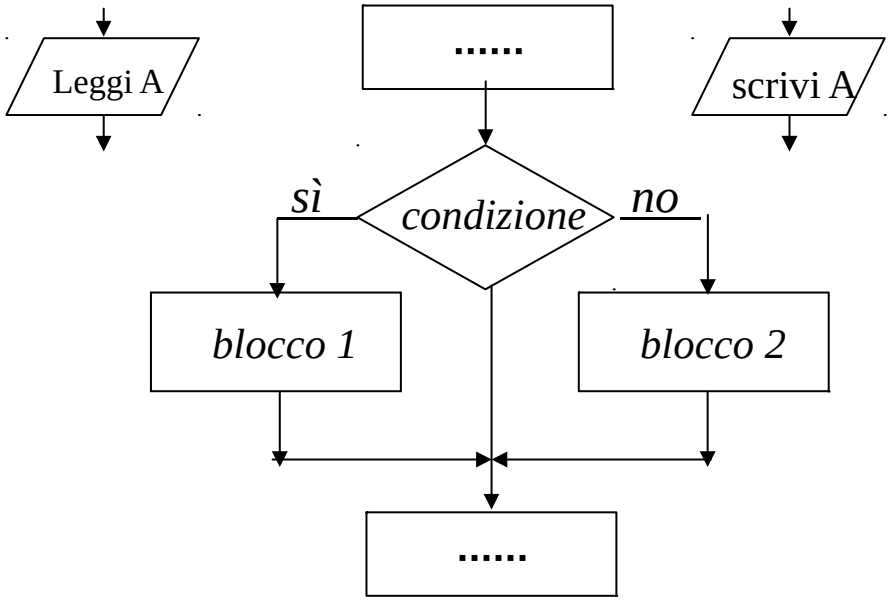
Il metodo dei **diagrammi a blocchi** consiste in una descrizione grafica; esso permette un visione immediata dell'intero procedimento e dell'ordine di esecuzione delle varie istruzioni.
I diagrammi a blocchi sono formati da simboli di forma diversa, ciascuna con un proprio significato; all'interno di ogni simbolo è presente un breve testo sintetico.
Linee orientate con frecce, che uniscono fra loro i vari simboli,



Assegnamento ed istruzioni aritmetico-logiche, Istruzioni I/O



Strutture di controllo: condizioni a due vie



4.4 Il concetto di variabile

Le variabili sono gli oggetti utilizzati dalle istruzioni del programma. Esse risiedono nella memoria dell'elaboratore e corrispondono a contenitori dei valori che sono elaborati durante l'esecuzione del programma. Alle variabili è associato un nome detto identificatore, che le individua univocamente all'interno del programma. Sono chiamate variabili perché a loro viene associato un valore che può cambiare durante l'esecuzione del programma.

I dati sono i i valori assunti dagli attributi degli elementi che caratterizzano il problema, rappresentati con variabili e costanti.

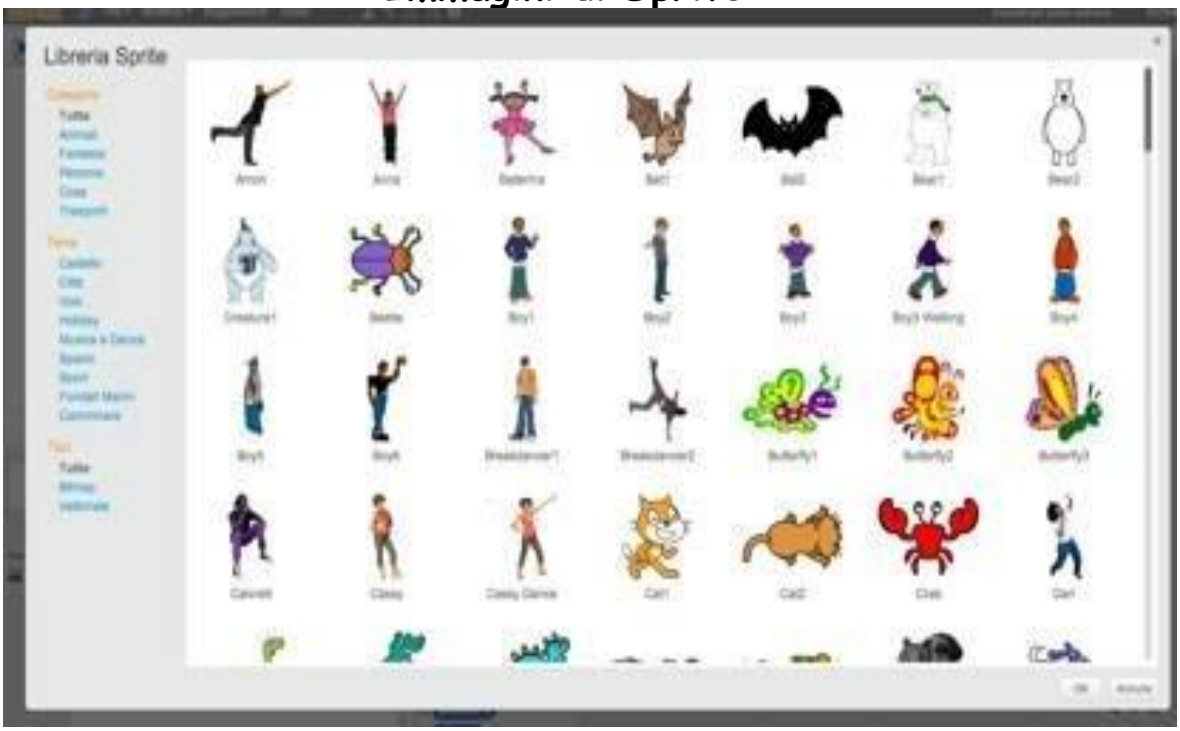
Le azioni sono le attività che mettendo i i dati in relazione tra loro, consentendo di ottenere i i risultati desiderati.
I dati possono essere:dati possono essere:
a) Elementari
numerici
alfabetici
alfanumerici
(Stringhe)
b) Non Elementari (insieme
di dati elementari)
Le azioni possono essere
riconducibili ad operazioni:
•• tipo aritmetico
•• tipo logico

4.5 I contenuti: Sprite, Sfondi e Costumi

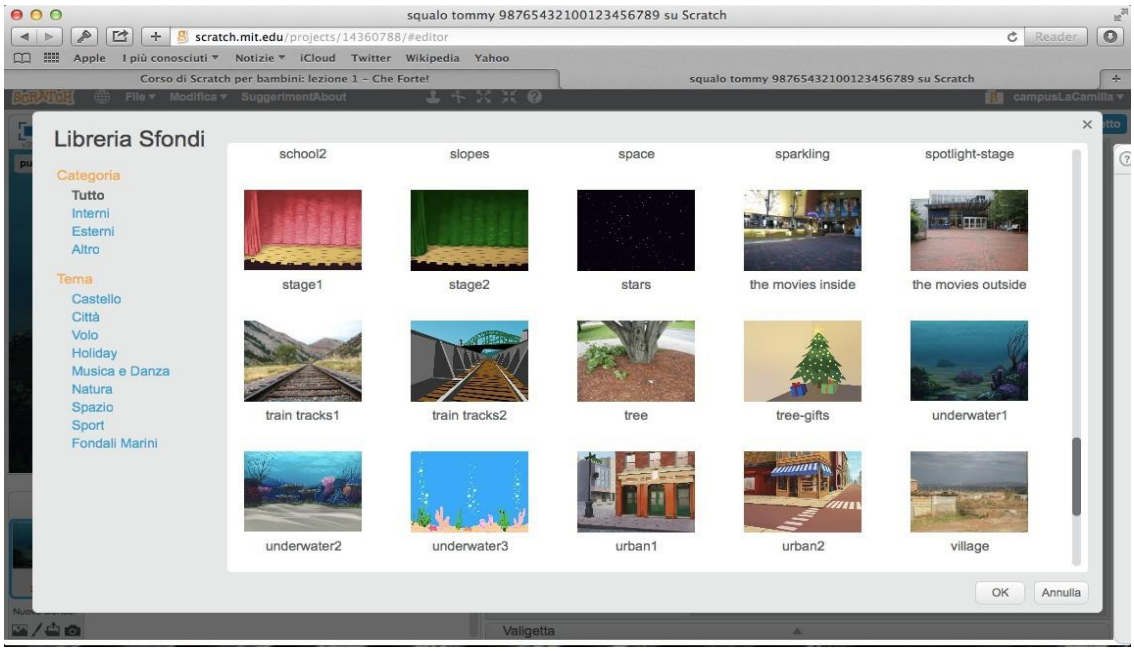
Scratch, la codifica per gioco: Lavorare con Scratch risulta molto più semplice e veloce. La codifica dei programmi in Scratch consiste nell'impilare blocchi grafici che presentano forma e colore dipendenti dall'istruzione che si vuole utilizzare, come si fa con i

mattoncini delle costruzioni. Il risultato del programma è visibile nell'area Stage (palcoscenico).
I Programmi di Scratch agiscono su oggetti grafici, disegni e immagini chiamati Sprite, come la figura del gatto. E' possibile disegnare degli Sprite a piacere attraverso un programma di disegno, così come è possibile importare immagini o una foto scattata con una macchina fotografica o con la webcam.
Gli Sprite si possono personalizzare associando loro Costumi diversi in modo da animarli con forme3 e suoni diversi. Ad ogni Sprite vengono associate le istruzioni che indicano cosa deve fare: parlare, muoversi, suonare, eseguire calcoli, ecc.

Immagini di Sprite



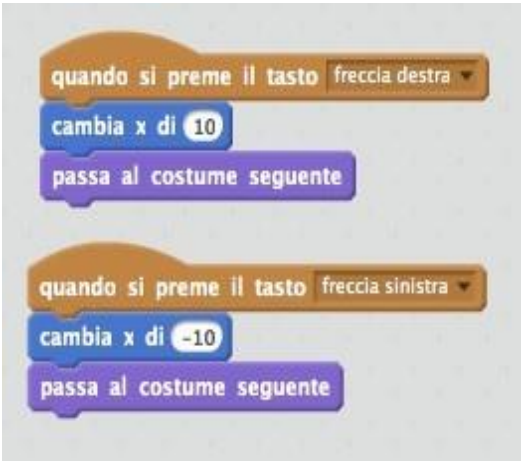
Immagini di Sfondi



Immagini di Costumi

Albero di Natale

Albero di Natale con stella cometa che si accende e si spegne



4.6 I contenuti: l'interfaccia di Scratch,spiegazione dei comandi

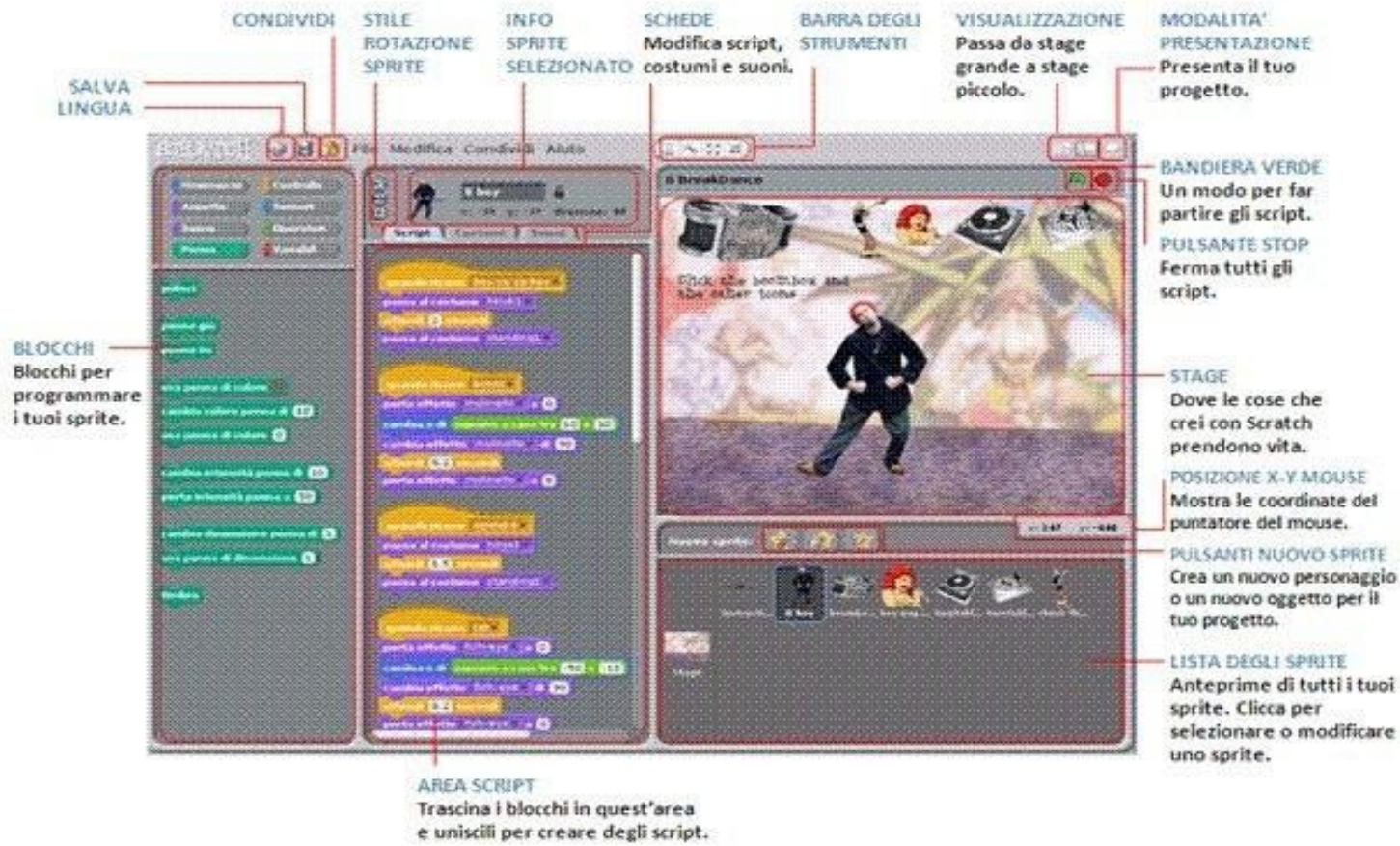
L'interfaccia di Scratch è suddivisa in 3 parti: un'area a destra detta Stage compare lo sprite del gatto; una centrale detta Script dove compaiono blocchi relativi alle istruzioni che permettono allo sprite (il gatto) di interagire. L'area a sinistra contiene i blocchi suddivisi in 10 categorie e che si distinguono anche per il colore:

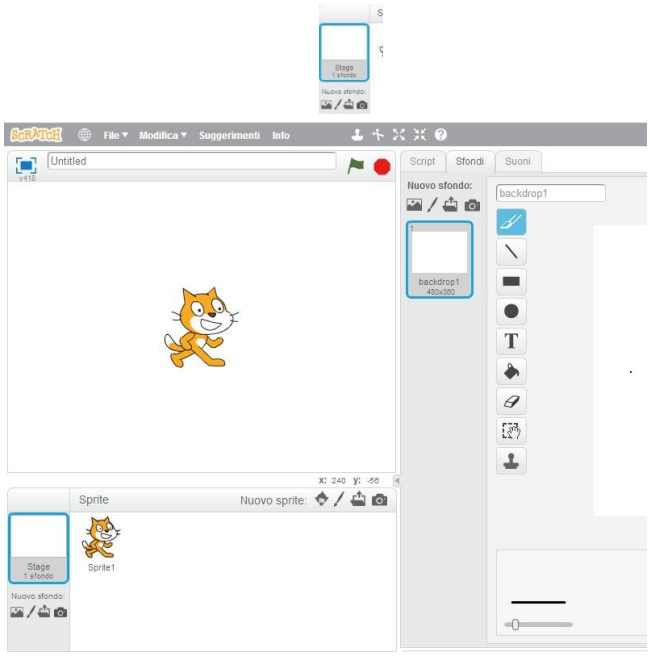
1- Movimento (blu): consente il movimento dello sprite (animazioni) rotazioni, scivolamenti, rimbalzi)

2-Aspetto (viola): è possibile cambiare dinamicamente l'aspetto dello sprite (ad esempio aumentando la dimensione e variando il colore)

3- Suono (lilla): offre suoni di strumenti musicali, voci, rumori, ed è possibile registrare voci e suoni e personalizzarli.

4-Penna (verde):è possibile costruire programmi di disegno con il computer in modo semplice ed intuitivo.





Cliccando sullo stage in basso a sinistra, i TAB posizionati in alto al centro, saranno relativi al solo stage. Abbiamo 4 modi di caricare uno sfondo e sono indicati dalle quatto icone sotto la scritta stage:

- Scegli uno sfondo dalla libreria
- Disegna un nuovo sfondo
- Carica uno sfondo da un file
- Nuovo Sfondo dalla webcam

Scelta la modalità su come inserire uno sfondo sullo stage abbiamo la possibilità di andare nelle sezioni indicate dai TAB

- Script – per assegnare delle istruzioni allo sfondo
- Sfondi – per inserire e/o modificare gli sfondi
- Suoni - per assegnare dei suoni agli sfondi, caricandoli da una libreria, o registrandoli o ancora importandoli da un file. Per riprodurli

A ogni sfondo inserito come anche ad ogni suono viene dato oltre al nome proprio anche un numero progressivo di inserimento.

Gli script relativi allo sfondo sono diversi. Tra questi proviamo lo script

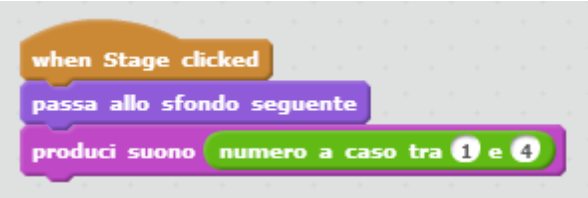
4.8 Esercitazioni (da proporre per casa)

1. Caricare 4 Sfondi e fare in modo che si alternino nella

visualizzazione con il click del mouse sullo stage stesso.

2. Inserire anche 4 suoni e fare in modo che vengano riprodotti in modo casuale ad ogni click e cambio di sfondo

Soluzioni:



Gli Sprite

Sono oggetti, disegni o comunque elementi presenti in un progetto scratch, che possono interagire con l'utente, tra di loro e con l'ambiente interno  o esterno a scratch stesso.
Cliccando sullo Sprite in basso a sinistra, i TAB posizionati in alto al centro, saranno relativi al solo sprite.
Abbiamo 4 modi di caricare uno  sprite e sono indicati dalle quatto icone sotto la scritta stage:

- Scegli uno sprite dalla libreria
- Disegna un nuovo sprite
- Carica uno sprite da un file
- Nuovo sprite dalla webcam

Scelta la modalità su come inserire uno sfondo sullo stage abbiamo la possibilità di andare nelle sezioni indicate dai TAB dove troviamo:

- Script – per assegnare delle istruzioni allo sprite – ogni sprite può avere istruzioni diverse
- Costumi – per inserire e/o modificare gli sprite
- Suoni  - per assegnare dei suoni agli sprite, caricandoli da una  libreria, o  registrandoli o ancora importandoli da un file

inserito uno sprite possiamo quindi andare nel TAB Script e assegnargli qualche comando.

Esercizio guidato:

posizioniamo lo sprite a sinistra sullo stage, assegnamo allo sprite il comando di movimento "fai 10 passi" quando si preme il tasto freccia a sinistra, oppure al posto dei 10 passi un numero di passi casuali da 0 a 10 mettendo al posto del "10" l'istruzione . Incastriamo i blocchi di istruzioni

nell'unico  modo possibile e verifichiamo lo script. Come vedremo a un certo punto lo sprite uscirà dallo  stage e non sarà più visibile. Per ovviare a questo  dobbiamo mettere nello script  relativo allo sprite in questione una ulteriore istruzione che impedisca allo sprite di uscire dallo stage. Avremo cosi il seguente blocco di istruzioni

Esercizio guidato:

Disegnamo una linea nera all'interno di uno sprite, e carichiamo un altro sprite con una figura qualsiasi.  Posizioniamolo al centro dello stage L'intento è di muovere il  secondo sprite verso la linea nera e di farlo  rimbalzare quando la tocca.

Facciamo fare quindi i passi al secondo sprite e se incontra la linea nera  facciamogli fare 20 passi indietro.

Esercitazioni:Esercitazioni:

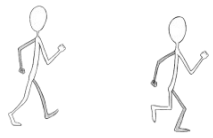
1) Caricare lo sprite disegnato in precedenza  duplicarlo nel TAB Costumi premedo il tasto sinistro del mouse e modificare il suo aspetto in  . Fatto ciò alternare i due costumi di questo sprite usando il blocco
2) Assegnare a questo sprite anche il movimento facendo in modo che lo sprite segua il mouse nei suoi movimenti sullo  stage tramite il blocco avendo cura di  incastrare le istruzioni in un ciclo

3.  Disegnare usando le varie forme nel paint editor di scratch.
4. Disegnare  usando solo il cerchio presente nel paint editor di scratch
5. Caricare come sfondo un bosco, inserire lo sprite disegnato a forma di uccellino e farlo muovere usando le istruzioni "fai passi" e "rimbalza"
6. Creiamo

I Costumi

Il costume è una caratteristica che ha lo sprite e che gli permette di cambiare look.

La forma degli sprite può essere cambiata durante l'esecuzione del



programma.
Per far si che questo avvenga, abbiamo a disposizione il Tab Costumi.
All'interno di questo TAB, proprio come all'interno del TAB sfondo visto in precedenza, abbiamo la possibilità di caricare, creare o modificare i costumi relativi ad uno sprite.

Conclusioni

Come adeguarsi a un mondo sempre più tecnologico, visuale, immediato, impattante cercando di rendere l'apprendimento "attivo"?
Come, quali procedure attuare per combattere efficacemente la dispersione scolastica?
Come coinvolgere gli alunni nella didattica in modo divertente e funzionale con le loro tipologie differenti di apprendimento?
Sono alcune delle tante domande che spesso ci poniamo come insegnanti che vogliono essere al passo con i cambiamenti che inevitabilmente hanno invaso il nostro fare tradizionale .
Forse la risposta è proprio in questa ultima frase, cambiare il "tradizionale" per "reinventare" e "reinventarsi" senza per questo sostituire quella didattica tradizionale che è e rimane quella solida impalcatura in cui regge tutto il nostro sapere, ma potrebbe essere utile per essere in linea con una generazione di studenti nati col digitale integrare tale sapere per far acquisire il saper fare. Il Coding è una base solida e fondamentale del mondo digitale "attivo" sulla quale integrare un nuovo modo di fare didattica.

Sabrina Fenu