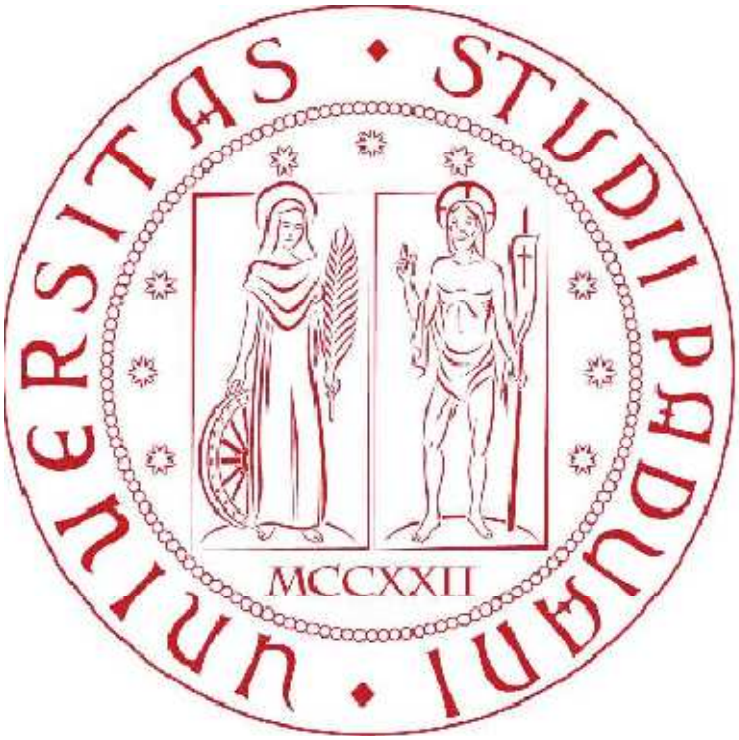
UNIVERSITÁ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
CORSO DI LAUREA IN INGEGNERIA INFORMATICA



PARADIGMI DI PROGRAMMAZIONE INTRODOTTI CON ALICE

Anno Accademico 2012/2013

Relatore:
Ch.mo Prof. Michele Moro

Laureando:
Enrico Massarente
Matr: 1006581

SOMMARIO

La seguente tesi ha lo scopo di illustrare il funzionamento di Alice, un software gratuito ed innovativo, ideato per gli studenti e per tutti coloro che vorrebbero imparare la programmazione orientata agli oggetti, pur non avendo conoscenze o prerequisiti di base. Si tratta quindi di un primo approccio alla programmazione vera e propria, che attraverso un ambiente di programmazione 3D rende facile creare l'animazione necessaria per raccontare una storia, giocare ad un gioco interattivo oppure un video da condividere nel web. Lo studente, attraverso la creazione di video animati o semplici videogiochi, è in grado di apprendere i concetti basilari ed i meccanismi che stanno dietro alla programmazione orientata agli oggetti.
L'interfaccia interattiva di Alice permette di effettuare operazioni come creare, spostare o modificare vari oggetti, operazioni alle quali corrispondono delle istruzioni in un linguaggio di programmazione orientato agli oggetti come ad esempio Java, C++ e C#. Alice permette di vedere immediatamente i cambiamenti prodotti dalle modifiche effettuate dall'utente all'interno di un programma, poiché ne mostra il funzionamento rendendo di più facile comprensione la relazione tra i costrutti e le istruzioni del programma ed il comportamento degli oggetti e dell'animazione del programma in esecuzione. Manipolando gli oggetti di questo mondo virtuale gli studenti sono quindi in grado di comprendere, imparare ed assimilare i concetti ed i costrutti basilari della programmazione, del pensiero computazionale e del problem solving. Quindi, attraverso l'analisi delle tre versioni in cui è stato ideato il software e delle metodologie per creare un programma, si riescono ad intuire le enormi potenzialità in campo educativo che tale software presenta, poiché permette di introdurre agli studenti fondamentali nozioni teoriche usando però un approccio più leggero e soprattutto più interessante.

INDICE

1 INTRODUZIONE

1.1 Obiettivi della tesi

Gli obiettivi per i quali questa tesi è stata realizzata sono molteplici. Innanzitutto si vuole descrivere il funzionamento del software Alice, per poter successivamente valutare l'efficacia didattica che il suo utilizzo può comportare. Quindi, attraverso l'analisi dei possibili vantaggi e svantaggi che uno studente può avere apprendendo le basi della programmazione mediante tale software, si vuole cercare di stabilire se tale strumento possa essere introdotto in un contesto scolastico o universitario per facilitare lo studio e migliorarne la qualità. Inoltre si vuole cercare di capire se esso potrà in futuro essere in grado di attirare un maggior numero di studenti, in particolare di sesso femminile (poiché tale categoria è sottorappresentata), verso l'informatica, scienza che molti ritengono troppo complessa e che per questo sono portati ad abbandonare o a non prendere nemmeno in considerazione per il loro percorso di studi.

1.2 Contesto operativo e strumenti utilizzati

L'analisi del software Alice è stata effettuata utilizzando una macchina con sistema operativo Windows XP, ma il software è disponibile per diversi sistemi operativi. Sono state testate le seguenti versioni: Alice 3.1.61.0.0, Alice 2.3b e l'ultima versione di Storytelling Alice risalente al 10 gennaio del 2007, poiché quest'ultimo software non è più supportato.

1.3 Prerequisiti

Per poter leggere e capire completamente questa tesi è necessario avere almeno le conoscenze basilari per quanto riguarda i linguaggi di programmazione orientati agli oggetti. In particolare una discreta conoscenza di Java sarebbe sufficiente a capire a pieno ogni argomento trattato. Infatti questa tesi non ha come scopo quello di spiegare i concetti fondamentali riguardanti un linguaggio di programmazione orientato agli oggetti, e quindi per alcuni argomenti vengono fatti solamente dei brevi richiami, dando per scontata la conoscenza degli altri da parte del lettore. In mancanza di tali conoscenze è comunque possibile comprendere molti dei concetti contenuti ed il funzionamento del software Alice, anche se non potrà essere del tutto chiara la corrispondenza con Java (o con altri linguaggi di programmazione) e con alcuni suoi concetti.

2 IL SOFTWARE ALICE

2.1 Introduzione

L’introduzione alla programmazione è sempre stata molto frustrante per gli studenti, molti dei quali spesso abbandonavano tale strada a causa della difficoltà. È quindi importante che uno studente abbia una prima esperienza più incoraggiante per quanto riguarda il mondo dell’informatica. Il sistema di Alice rappresenta una svolta nella didattica per quanto riguarda l’insegnamento della programmazione orientata agli oggetti: gli oggetti in Alice sono rappresentati sotto forma di persone, animali e cose che popolano la realtà virtuale ed il cui stato viene modificato mediante metodi con nomi quali per esempio "sposta in avanti di un metro" o "ruota a sinistra di un quarto di giro" che sono facilmente ed intuitivamente comprensibili per gli studenti. Uno dei punti di forza di Alice è che è stato in grado di rendere per la prima volta concreti agli occhi dei programmatori dei concetti che prima erano astratti, come ad esempio il cambiamento di stato di un oggetto che ora viene mostrato visivamente.
Un altro punto a favore per Alice è la capacità di motivare gli studenti ad apprendere, non con le più classiche delle motivazioni quali premi o punizioni, bensì con la programmazione di narrazioni, attività universalmente convincente poiché alla maggior parte delle persone piacerebbe creare una propria storia. L’interfaccia di Alice fornisce un ambiente di programmazione “drag and drop” ben ingegnerizzato che permette agli studenti di trascinare i componenti del programma per lo schermo e garantisce che lo studente non possa fare errori di sintassi (può invece commettere errori logici, cioè il programma non fa ciò che si vorrebbe o fa qualcosa di sbagliato), cosa che di solito è molto frustrante. Nonostante ciò si sviluppa però un’intuizione per la sintassi da parte dello studente, anche perché vi è la possibilità di visualizzare il codice del proprio programma con una sintassi simile a quella di Java.

2.2 Problematiche nell’insegnamento della programmazione

Alice è stato sviluppato per affrontare diversi problemi fondamentali che si presentano nell’insegnamento della programmazione.

Molti linguaggi di programmazione sono stati progettati per “produrre codice”, e ciò introduce ulteriore complessità e contribuisce ad allontanare gli studenti. Infatti, sebbene il problema principale nella scrittura del codice sia la difficoltà nel creare un programma, spesso lungo, che sia coerente e funzionante in tutte le sue parti, non è da sottovalutare “l’impressione” che tale attività può dare vista dall’esterno. Rimanere per molte ore concentrati davanti ad un computer scrivendo con la tastiera, per molti non è una prospettiva allettante, specie per quanto riguarda l’universo femminile, e contribuisce alla creazione di stereotipi nei confronti dei programmatori che spingono ulteriormente gli studenti a non scegliere l’informatica nel loro corso di studi. Questo purtroppo, con la diminuzione degli iscritti ai corsi di studi relativi all’informatica, non è un fatto irrilevante come si potrebbe pensare, e quindi non va assolutamente trascurato. È quindi indispensabile saper dare agli studenti alcune ragioni che spingano a voler scrivere un programma per computer, analizzare ed affrontare i timori che si possono avere sulla programmazione. Alice risolve subito alcuni di questi problemi che affliggono i programmatori alle prime armi: non si deve usare la tastiera per scrivere codice come nei classici linguaggi di programmazione ma è sufficiente l'uso del mouse grazie all'ambiente “drag and drop” che permette di trascinare le righe di codice nell’editor appropriato e successivamente vedere l’esecuzione del programma. Il programmatore quindi diventa quasi come un coreografo o un regista che decide cosa devono fare i personaggi che egli stesso ha creato. E quella della creazione di un racconto risulta un’attività piuttosto stimolante per tutti, sia per gli uomini che per le donne, avendo quindi il potenziale di aumentare l’interesse nei confronti di questa disciplina.

Inoltre Alice è stato progettato unicamente per insegnare la teoria della programmazione senza la complessa semantica di molti linguaggi di programmazione come C, C++ e Java che spesso spaventa gli studenti e li spinge ad abbandonare lo studio dell'informatica. Gli utenti possono spostare oggetti dalla galleria di Alice al mondo virtuale che essi hanno immaginato, e poi è possibile programmare trascinando riquadri che contengono strutture logiche. Inoltre, l'utente può manipolare la telecamera di Alice e l'illuminazione per apportare ulteriori miglioramenti. Infatti Alice è congiunto con il suo ambiente di sviluppo integrato (IDE). Quindi, non solo non esiste una sintassi da ricordare, ma per come è stato progettato, non è nemmeno possibile commettere errori di sintassi, ed eventuali errori (non quelli logici) vengono tempestivamente segnalati da Alice non appena commessi. Ciò contribuisce efficacemente a ridurre la frustrazione dovuta alla ricerca e alla correzione degli errori sintattici. Tuttavia Alice supporta completamente la programmazione orientata agli oggetti e permette di visualizzare il codice anche con una sintassi molto simile a quella di Java. Alice ha quindi le potenzialità per attrarre quella parte della popolazione normalmente non esposta alla programmazione, come ad esempio gli studenti di sesso femminile in età di scuola media, incoraggiando la narrazione, a differenza di molti altri linguaggi di programmazione che sono progettati per il calcolo. Alice è utilizzato anche in molti college ed università come introduzione ai corsi di programmazione.

Un altro problema che si presenta per tutti quegli studenti che sono abituati a programmare in maniera classica è che avendo a che fare con del codice astratto, non sempre ci si rende conto di cosa effettivamente si sta facendo, dell'utilità, degli effetti e dei cambiamenti che la scrittura di tale codice comporta concretamente. Quindi uno degli scopi di Alice è anche quello di rendere possibile a tutti la rapida creazione di una propria realtà virtuale, sebbene non graficamente dettagliata, piuttosto semplicistica e senza tanti tecnicismi, che però risulta visibile e manipolabile dallo studente diventando per egli qualcosa di concreto. Infatti l'utente si rende

immediatamente conto del suo operato, senza perdere tempo, e riesce a vedere distintamente le modifiche che ha apportato, apprendendo velocemente cosa comportano nella sua realtà virtuale, i cambiamenti effettuati.

2.3 Le varie versioni di Alice: usi e differenze

Un lettore curioso probabilmente si sarà chiesto a cosa sia dovuto il particolare nome scelto per il software. Alice non è un acronimo: non significa nulla, è semplicemente un nome. Il software si chiama così in onore di Charles Lutwidge Dodson, un logico e matematico inglese che scrisse, sotto lo pseudonimo di Lewis Carroll, “Alice’s Adventures in Wonderland” e “Through the Looking Glass”. Lewis Carroll era un matematico, scrittore e fotografo. Proprio come le persone che hanno costruito Alice, egli è stato in grado di fare della matematica e della logica complessa, ma riteneva che la cosa più importante fosse quella di rendere la sua disciplina semplice ed affascinante per ogni suo apprendista. Quindi pur potendo fare cose intellettualmente molto difficili, realizzò che la cosa più significativa fosse quella di essere in grado di comunicare con chiarezza ed in modo divertente. Della stessa idea sono ora i creatori di Alice, che hanno voluto seguire il suo pensiero creando uno strumento in grado di facilitare ed agevolare gli studenti nell’apprendimento della programmazione, rendendo quindi divertente e molto più interessante qualcosa di così complesso.

Il software fu inizialmente sviluppato all’Università della Virginia, e successivamente, dal 1997, alla Carnegie Mellon University da un gruppo di ricerca guidato dal compianto Randy Pausch. Il progetto di Alice si sforza di esporre gli studenti alla scienza informatica con l'introduzione di un linguaggio di programmazione “user-friendly”. Il progetto è stato inizialmente finanziato dalla National Science Foundation ed è stato portato avanti in sei regioni: Durham nel Nord Carolina, Virginia Beach in Virginia, San Jose in California, Denver in Colorado, Charleston nella Carolina del Sud, ed Oxford in Mississippi, dal 2006 al

2009. A partire da giugno del 2011, la Duke University ha ricevuto una sovvenzione di 2,5 milioni di dollari da parte della National Science Foundation per introdurre Alice nelle aule scolastiche in tutta la Carolina del Nord e del Sud e nel Mississippi.

Susan Rodger, professore del dipartimento di scienze informatiche alla Duke University e leader regionale del progetto ritiene che Alice possa essere uno strumento importante per far sì che l'informatica diventi una tra le materie principali insegnate e scelte dagli studenti, cosa che al momento non avviene in maniera abbastanza soddisfacente. Grazie alle sue componenti narrative, Alice può anche essere un ottimo modo per coinvolgere più ragazze nel campo prevalentemente maschilista dell'informatica. L'obiettivo principale di questo progetto è quello di coinvolgere gli studenti a partire dalle scuole medie ed introdurli al mondo dell'informatica, in modo tale che poi possano scegliere l'informatica trai corsi che svolgeranno alla scuola superiore o all'università.
Dal 2008 il team di Alice ha formato più di 120 insegnanti ed ogni estate gestisce dei laboratori che consentono di imparare come utilizzare Alice ed integrarlo nel programma delle loro classi.

Nel corso degli anni, il software è stato sviluppato ed aggiornato in diversi modi, ed attualmente è conosciuto con tre nomi diversi e versioni diverse: Alice 3.x, Alice 2.x e Storytelling Alice.
All'avvio del progetto di Alice, gli sviluppatori avevano pensato inizialmente ad Alice 3 come una versione successiva, che avrebbe sostituito Alice 2. Ma questo pensiero si è ben presto evoluto ragionando su come e per quali scopi le persone utilizzavano Alice. Infatti il software avrebbe potuto essere creato con varie funzionalità, garantendo quindi un certo grado di completezza e la capacità di svolgere molteplici compiti, ma probabilmente non li avrebbe svolti tutti in modo ottimale. Focalizzandosi su determinati compiti in un ambito più ristretto è invece possibile ottenere un software qualitativamente migliore, sebbene limitato nei compiti che è in grado di svolgere. È per questo che si è deciso di mantenere separati Alice 3

ed Alice 2: si tratta infatti di strumenti diversi che hanno scopi diversi, e la percezione (dovuta alla numerazione) che uno sostituisca l'altro è perciò sbagliata. Inoltre essi sono due software diversi perché fanno anche riferimento ad utenti diversi in base ad età, capacità, livello, risultati ed obiettivi che si vogliono raggiungere. Dal punto di vista pratico, il mantenimento di entrambe le versioni è dovuto anche al fatto che il materiale per Alice 3 è ancora in fase di costruzione, mentre quello per Alice 2 è molto più vasto e sono inoltre presenti dei libri di testo per facilitarne l'insegnamento e l'utilizzo. Infatti, come è stato anche notato durante la stesura di questa tesi, per uno studente principiante, avere a che fare con un software non accompagnato da un libro di testo adeguato è un problema da non sottovalutare. Per coloro che hanno già esperienza nel campo della programmazione, la comprensione di Alice 3 (per cui non vi sono ancora libri di testo) risulta piuttosto semplice ed intuitiva. È chiaro però che il software è rivolto a degli studenti alle prime armi, e non avere del materiale adeguato su cui studiare può comportare maggiori difficoltà nell'apprendimento ed uno sforzo maggiore da parte dei docenti nell'insegnamento. Inoltre per quanto riguarda Alice 3, per ora non sono ancora stati creati dei tutorial, che invece sono presenti nelle versioni precedenti e risultano estremamente utili per introdurre l'interfaccia allo studente e fargli prendere direttamente confidenza con il software dal punto di vista pratico piuttosto che concentrarsi per ore solo sull'apprendimento della teoria.

Sebbene Alice 2 e 3 siano abbastanza simili per molti aspetti, essi sono stati ideati per scopi diversi e per utenti diversi. Alice 3 è stato specificatamente disegnato per essere usato in corsi nei quali gli studenti dovranno successivamente imparare concetti più avanzati ed essere in grado di scrivere codice Java alla fine del corso. In particolare è rivolto agli studenti delle scuole superiori ed universitari. Per quanto riguarda Alice 2, esso vuole essere una prima introduzione ai concetti della programmazione, senza per forza dover essere integrato con l'insegnamento della programmazione vera e propria e di linguaggi orientati agli oggetti come Java. Quindi attualmente viene mantenuto distinto rispetto all'ultima versione anche perché è rivolto a studenti delle

scuole medie, o al limite degli ultimi anni delle elementari. Tuttavia nei primi anni dello sviluppo del software, non esistendo ancora la terza versione, è stato utilizzato con gli stessi scopi che ora sono relativi ad Alice 3, ottenendo comunque un discreto successo. Inoltre, in mancanza di un testo, del materiale e di una galleria di oggetti adeguata per l'ultima versione, per il momento è forse preferibile continuare ad utilizzarlo per l'insegnamento universitario, attendendo sviluppi futuri. Infatti come già detto, le differenze e le lacune per alcuni aspetti rispetto ad Alice 3, non sono ancora tali da non poter essere colmate da un docente capace o da impedirne l'adozione in un contesto universitario.

È comunque utile elencare alcune delle altre differenze tra Alice 2 e 3 per comprendere anche meglio la scelta, fatta in questa tesi, di descriverle entrambe. Innanzitutto vi sono alcune proprietà di default che sono impostate in maniera diversa nei due ambienti (per esempio vi è un'impostazione differente della proprietà "vehicle", descritta più avanti). Ovviamente le impostazioni iniziali si possono cambiare ma bisogna sempre tenere presente che ci si sta rivolgendo ad un pubblico inesperto che potrebbe avere qualche difficoltà.

Alice 2 non fornisce una completa implementazione del concetto di ereditarietà. Quando una nuova classe viene creata, si ottiene una copia delle proprietà e dei metodi della classe base e viene salvata in un nuovo file di modello 3D. Modifiche successive alla superclasse non si riflettono nella sottoclasse. Questo probabilmente perché si vuole che l'ereditarietà venga trattata solo più avanti, magari in un corso successivo, quando gli studenti avranno raggiunto una certa padronanza con il software ed assimilato alcuni concetti fondamentali a livello teorico. Inoltre in Alice 2 non vengono aperte e visualizzate, nell'editor, le varie classi, per quanto riguarda gli oggetti o la scena, cosa che invece si può fare in Alice 3 (cioè si possono aggiungere metodi a tali classi, ma in nessuno dei due software è comunque possibile accedere, visualizzare o modificare il codice pre-scritto dagli sviluppatori per quanto riguarda le classi di oggetti), rendendo quindi il concetto di ereditarietà meno evidente ed intuitivo da comprendere. Confrontando poi le modalità relative alla

visualizzazione del codice con una sintassi simile a quella di Java, si nota che Alice 2 mostra meno dettagli sintattici, mentre il codice mostrato da Alice 3 è più simile al vero e proprio codice Java e più accurato. Perciò, la maggior completezza di Alice 3, rende molto più facile il passaggio da esso ad un linguaggio di programmazione orientato agli oggetti, rispetto ad Alice 2. Risulta quindi anche più semplice effettuare dei collegamenti a livello concettuale tra gli argomenti trattati.
Per questi motivi per il momento le due versioni vengono mantenute separate, sono entrambe supportate e vengono aggiornate in parallelo.

Una variante di Alice 2, chiamata Storytelling Alice è stata creata da Caitlin Kelleher (uno degli sviluppatori di Alice) per la sua tesi di dottorato. Essa presenta alcune principali differenze. Innanzitutto è molto più incentrata sulla produzione di storie sensate e complete, e questo comporta dei notevoli cambiamenti per quanto riguarda gli elementi e la struttura del programma. È stata modificata l'interfaccia in modo da poter creare e gestire contemporaneamente più di una scena proprio come nelle vere storie, cosa non possibile nelle altre versioni. Sono stati aggiunti pulsanti per rendere più intuitivo e semplice l'uso del software. Sono stati creati nuovi tutorial focalizzati maggiormente sulla creazione di racconti. Sono stati aggiunti molti più elementi, paesaggi e personaggi 3D con animazioni personalizzate progettate per stimolare idee per le storie poiché si vuole che l'utente abbia a disposizione tutto il materiale necessario per la realizzazione del racconto che egli ha immaginato (sebbene sia impossibile riprodurre tutto ciò che una persona è in grado di immaginare). Sono stati inoltre modificati e creati nuovi metodi per i personaggi, ed in particolare i più comuni e naturali che si possano pensare per creare un racconto. Tutte queste modifiche sono state apportate perché Storytelling Alice è rivolto soprattutto ad un pubblico femminile, in particolare alle studentesse della scuola media, per cercare di attrarle verso l'informatica, disciplina prevalentemente maschilista. Infatti non si pone come obiettivo quello di insegnare loro i concetti teorici fondamentali della programmazione, poiché molto probabilmente fallirebbe vista la riluttanza che hanno

le ragazze, specie di così giovane età, verso questo vasto campo.
Anche per via del fatto che è stato sviluppato come progetto per una tesi, il software Storytelling Alice non è più supportato dal 2007, anche se rimane tuttora disponibile sul sito di Alice per via delle numerose richieste, e per esso non sono mai stati realizzati libri di testo o altri materiali didattici. Tuttavia, prendendo spunto da tale progetto, si sta cercando di creare una versione successiva di Storytelling Alice, per ora conosciuta con il nome di Looking Glass (che deriva anch'esso da un'opera di Lewis Carroll) ed attualmente in fase di sviluppo presso la Washington University di Saint Louis.

3 ALICE 3.X

3.1 Introduzione

In questa sezione si vuole descrivere la nuova versione del software, cioè Alice 3, ideata per essere in futuro utilizzata nelle scuole superiori o nelle università, come introduzione ai corsi di programmazione relativi ai linguaggi di programmazione orientati agli oggetti. Tale versione, oltre ad un'interfaccia rinnovata rispetto alle precedenti (anche gli oggetti sono decisamente diversi), presenta delle sostanziali modifiche come la possibilità di vedere la gerarchia generale di classi del programma ed apportarvi dei cambiamenti. Anche da questo si comprende come l'obiettivo principale del software sia quello di far apprendere i fondamenti teorici della programmazione in maniera più semplice, intuitiva ed efficace. Rispetto alle altre versioni infatti vi è meno interesse riguardo alle storie che vengono create, poiché esse non sono significative: non è necessario che i racconti siano perfetti, lunghi e dettagliati, ma che lo studente capisca ciò che sta facendo e riesca ad effettuare gli opportuni collegamenti tra il suo operato in Alice ed i concetti teorici in Java, apprendendo almeno in parte alcune regole sintattiche in esso presenti.

3.2 Descrizione dell'interfaccia

Una volta avviato Alice 3, comparirà una finestra di benvenuto (figura 3.1) che ha lo scopo di aprire un progetto su cui lavorare. Questa prima indicazione è fondamentale poiché permette ad Alice di mostrare una scena: se non vi sono progetti personali salvati si può scegliere tra alcune scene di base rappresentanti dei possibili sfondi, ovvero l'ambientazione per il video o il gioco che si vuole creare.

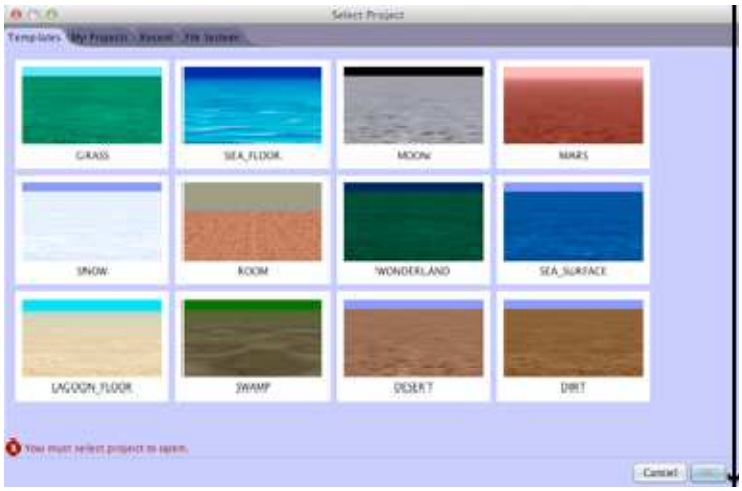


Figura 3.1: Scelta del progetto da aprire

Dopo aver selezionato la scena, essa viene mostrata in alto a sinistra dello schermo come si vede in figura 3.2. Infatti si può notare subito che l'interfaccia di Alice è un ambiente di programmazione costituito sia da un mondo virtuale popolato da personaggi ed oggetti, sia da un programma composto da istruzioni che rendono possibile il movimento e le azioni dei personaggi. In questo modo vi è una forte interazione tra il programmatore ed Alice, agevolata dalla facilità di utilizzo del programma. Nella parte destra della finestra iniziale vi è il "Code Editor Panel" in cui vengono create le diverse parti del programma (classi e metodi). Inoltre vi sono in basso a sinistra il "Methods Panel" ed in basso a destra il "Control Tiles Panel".

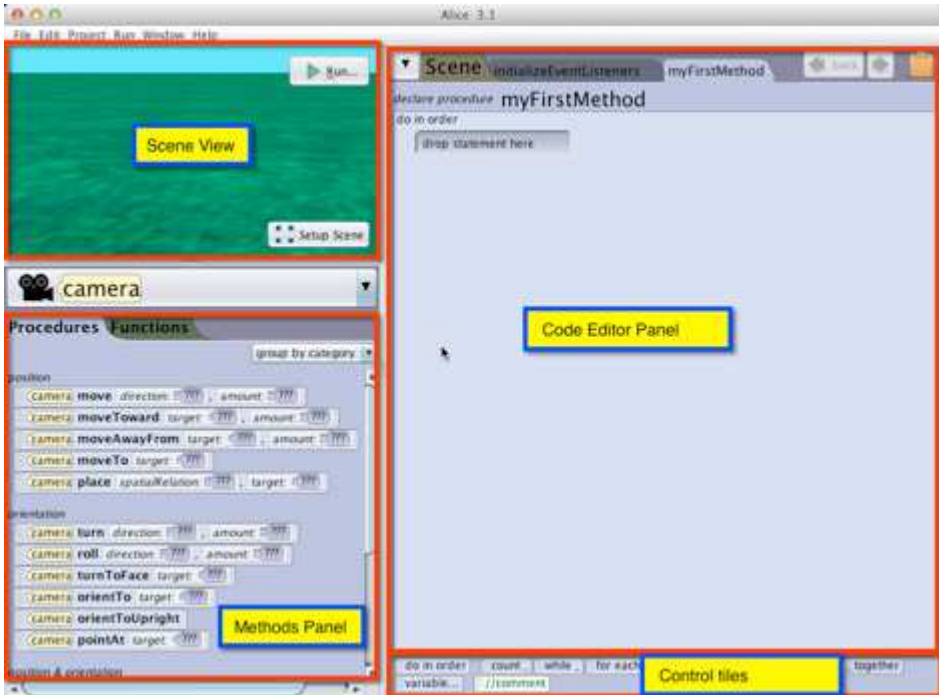


Figura 3.2: Finestra iniziale

Quando in Alice viene aperta per la prima volta una delle scene di default, “camera” è l’elemento selezionato, viene mostrata la scena selezionata e nel Code Editor Panel viene mostrato il metodo principale definito per la scena (chiamato “myFirstMethod”). Nel Methods Panel in ogni riga sono rappresentati i vari metodi (evidenziati in grassetto) che indicano le azioni fatte su o da una oggetto (persona, animale, cosa o altra entità) in una scena.

Come mostrato in figura 3.3 nel Methods Panel i metodi sono divisi in procedure (metodi che servono per fare un’azione) e funzioni (metodi che servono per fare una domanda). Nell’esempio sono mostrate delle procedure per la telecamera utilizzate per muoverla. Nel Control Tiles Panel vi sono degli elementi che servono per gestire ed organizzare istruzioni ed informazioni: per esempio specificano l’ordine in cui eseguire i metodi, in quali circostanze ripeterli (cicli) e se eseguirli (“if”). Vi è inoltre la possibilità di inserire commenti o variabili per contenere dei dati. Il Code Editor è un ambiente “drag and drop” il cui utilizzo è abbastanza intuitivo: invece di scrivere

“a mano” il codice, è sufficiente trascinare con il mouse i vari elementi desiderati dal Methods Panel e dal Control Tiles Panel e metterli nella posizione desiderata nel Code Editor, creando così le varie istruzioni del programma che permetterà l’animazione degli oggetti nella scena.

Figura 3.3: Metodi: Procedure e Funzioni

Prima di iniziare a scrivere il codice è necessario selezionare gli elementi, gli oggetti ed i personaggi desiderati che animeranno la scena, e ciò è possibile accedendo allo “Scene Editor Panel” (figura 3.4), tramite il pulsante “Setup Scene” che si trova nello “Scene View” (figura 3.2).

Figura 3.4: Scene Editor Panel

Come mostrato in figura 3.4, Scene Editor Panel è composto da due pannelli: “Scene Setup Panel” e “Gallery Panel”. Il primo presenta dei pulsanti per posizionare come meglio si crede gli oggetti nella scena e cambiare loro dimensione, colore o altre proprietà; il secondo invece è una vasta galleria contenente modelli 3D, ovvero gli oggetti e personaggi da inserire nella scena, divisi in varie classi e secondo temi. Se si desidera tornare al Code Editor Panel è sufficiente premere il pulsante “Edit Code” nel Scene Setup Panel.

È utile inoltre analizzare alcune delle opzioni presenti nella barra del menù (evidenziata in figura 3.5), tralasciando le più comuni e concentrando l’attenzione su quelle proprie di Alice.

Figura 3.5: Barra del menù

Il menù “File” contiene opzioni per gestire e creare i progetti, comuni a molti programmi e ben note. Bisogna porre solo attenzione all’opzione “Revert” poiché riporta il programma allo stato originale, cioè al momento in cui è stato aperto o all’ultimo salvataggio. All’interno del menù “File” vi è il menù “Print”, il quale consente di stampare tutto o parte del codice creato, escluso il codice pre-scritto e contenuto nel sistema di Alice (che non può nemmeno essere visualizzato o modificato). Nel menù “Edit” vi sono comandi ben noti come “Undo”, “Cut”, “Copy”, “Paste”, ma gli ultimi tre non sono stati implementati nella versione testata in questa tesi poiché non necessari per via dell’ambiente “drag and drop” che è Alice.
Il menù “Project” è composto dalle opzioni “Manage Resources”, “Find”, “Statistics”. La prima consente di importare e usare risorse come immagini e musica. La seconda permette di trovare, all’interno del programma, le parti in cui sono usati elementi del Control Tiles Panel ed inoltre, non appena si seleziona il metodo relativo alla scena, vengono anche aperti il metodo “main” e la classe “Program”.
La terza opzione invece fa comparire una finestra in cui sono riportati i metodi ed i costrutti con l’indicazione del numero di volte in cui sono richiamati nel programma.
Il menù “Window” è composto da “Perspectives”, “Project History”, “Memory Usage”, “Preferences”, elementi che servono per controllare ciò che è mostrato dall’interfaccia di Alice in termini di numero di finestre aperte e loro contenuto.
Il menù “Perspectives” permette di passare dal Code Editor al Scene Editor Panel, ed è quindi alternativo all’uso degli appositi pulsanti. L’opzione “Project History” mostra in ordine cronologico tutte le operazioni svolte sulla scena durante la sessione attuale come ad esempio la creazione o la modifica di oggetti. Inoltre questa opzione è molto importante poiché permette di ripristinare uno stato precedente all’interno della stessa sessione: selezionando un’azione tra la prima e la penultima, tutte quelle successive verranno eliminate dalla scena e rimarranno visibili (non più evidenziate) nella finestra, permettendo così all’utente di ripristinarle, solo fino alla prossima modifica effettuata, poiché essa comporta l’aggiornamento della cronologia.
Nell’opzione “Memory Usage” è possibile visualizzare un grafico che riporta

l’andamento dell’uso della memoria, con la possibilità di utilizzare il “garbage collector” di Java.
Molto ampio ed importante è invece il menù “Preferences” (figura 3.6), che consente di impostare le preferenze per l’ambiente di Alice. È possibile infatti scegliere il linguaggio di programmazione, cioè se visualizzare il codice con una sintassi simile a Java oppure con quella di Alice. Per chi non ha mai utilizzato un linguaggio di programmazione la prima scelta potrebbe essere utile per iniziare ad imparare la sintassi tipica di un linguaggio di programmazione come Java, mentre la seconda scelta ha il vantaggio di semplificare la lettura del codice.
Poi vi sono opzioni classiche e ben note ed altre a cui prestare più attenzione. Infatti è possibile scegliere di visualizzare: le classi nel Code Editor (molto utile per iniziare a comprendere il funzionamento di Java e concetti come l’ereditarietà), la parola “this” nel Code Editor (che si riferisce alla scena o all’oggetto selezionato ed è un termine del linguaggio Java), il tipo per le variabili, la classe “Program” che contiene il metodo “main” (che è il primo ad essere eseguito non appena viene premuto il pulsante “Run”).
È inoltre possibile abilitare: l’uso dei costruttori nelle classi (cioè dare la possibilità di visualizzare il codice contenuto nei costruttori e di modificarlo in parte), l’uso della ricorsione (strumento molto potente in informatica ma spesso fonte di errori se non compreso bene, motivo per cui viene aperta una finestra con ulteriori spiegazioni quando si tenta di abilitarlo), l’uso del valore “null” per le variabili locali o di classe (permette di inizializzare le variabili al valore “null”).
Vi sono infine delle preferenze nel menù “Gallery” relative alla finestra che appare quando viene aggiunto un nuovo oggetto alla scena.
Quindi è facile comprendere come una delle cose fondamentali che deve fare uno studente alla prima esperienza con un linguaggio di programmazione sia impostare correttamente le preferenze in modo da “personalizzare” l’ambiente di Alice ed adattarlo alle sue esigenze e capacità.

Non appena viene installato Alice, vengono impostate automaticamente le preferenze di default (figura 3.6).

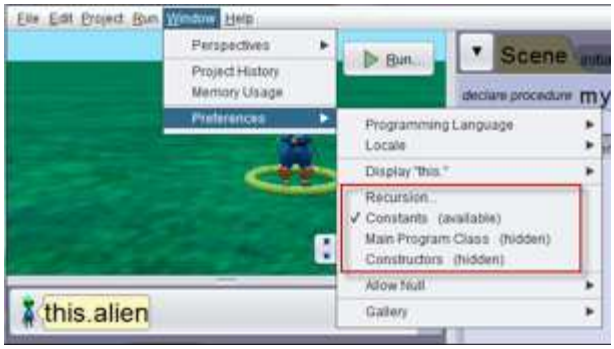


Figura 3.6: Preferenze

Molte delle opzioni descritte sopra sono disabilitate, l'unica abilitata è "Constants" che permette di creare delle variabili che sono costanti, cioè hanno un valore che non cambia durante l'esecuzione del programma. Con le preferenze di default nel Code Editor è evidenziato il metodo "myFirstMethod", contenuto nella classe "Scene". Dopo che il codice è stato scritto, è sufficiente cliccare sul pulsante "Run" per eseguirlo e vedere la corrispondenza tra le istruzioni in esso riportate e l'evolversi della scena. Per tutti coloro che intendono utilizzare Alice come strumento per prepararsi ad apprendere un linguaggio di programmazione orientato agli oggetti, le opzioni che conviene abilitare sono "Constants" e "Constructors". Potrebbe inoltre essere conveniente abilitare nel Code Editor la visualizzazione di tutte le classi compresa la classe "Program". Da notare che la classe "Scene" è sempre presente, mentre la presenza di ulteriori classi è dovuta al tipo di oggetti inseriti nella scena. Quando una classe viene selezionata essa è aperta nel Code Editor, in cui vengono mostrati i metodi in essa contenuti, divisi in tre gruppi: procedure, funzioni e proprietà. Nell'esempio riportato in figura 3.7 è stata aperta la classe "Alien" ed è stata abilitata anche l'opzione relativa ai costruttori (ed infatti compare la voce "constructors"). È ovviamente possibile aggiungere e creare le proprie funzioni e procedure per un determinato oggetto. Va ricordato che un costruttore è un metodo

speciale che contiene istruzioni che consentono di creare un nuovo oggetto di una determinata classe. Inoltre quando una classe viene mostrata, in basso a sinistra il Methods Panel viene sostituito con un diagramma che mostra la posizione della classe nella gerarchia delle classi del programma (cosa molto utile per iniziare a comprendere il significato di ereditarietà).

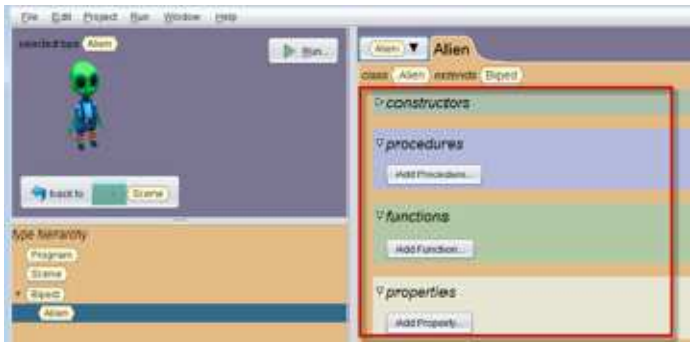


Figura 3.7: Metodi nelle classi

Come già detto in precedenza, una delle parti del Scene Editor Panel è costituita dal Gallery Panel (figura 3.8), dove sono contenuti i vari oggetti che si possono inserire nella scena. Sebbene questa parte possa in un primo momento sembrare marginale per l'apprendimento di un linguaggio di programmazione, è invece estremamente importante soffermarsi un attimo sul motivo per cui si è scelto di dividere gli oggetti in varie classi e categorie. Infatti ciò non è stato fatto solo per facilitare la ricerca delle figure desiderate da inserire nella scena, ma anche per far intuire all'utente la potenza di un linguaggio di programmazione orientato agli oggetti: il fatto che vari oggetti (ciascuno avente la propria classe) facciano parte di un'unica superclasse è in grado di far capire, o almeno intuire, anche ad un programmatore alle prime armi, che essi hanno delle caratteristiche comuni. Una volta compreso ciò è anche facile capire che molti metodi, funzioni e parti di codice sono in comune tra questi oggetti (che sono quindi delle sottoclassi) e possono essere utilizzate a proprio piacimento per specificarli al meglio o semplicemente riutilizzate senza doverle riscrivere poiché identiche a quelle della classe superiore.

La relazione tra le classi ed i personaggi della scena può essere quindi compresa meglio in questo modo: in Alice, una classe raggruppa la rappresentazione digitale di un'entità, un piano per costruirla ed istruzioni per animarla; un'ulteriore definizione generale è: una classe definisce il tipo di un oggetto (modello 3D) e le azioni che possono essere compiute da esso. Infine la galleria contiene le classi per creare ed animare gli oggetti nel mondo virtuale di Alice.
È da notare inoltre che se non vengono inseriti oggetti nella scena, la scena presenta comunque uno "sfondo" di base scelto all'inizio ed una telecamera, che sono entrambi delle classi. In particolare ogni oggetto della scena non è altro che un'istanza di una classe, la scena stessa è un'istanza della classe "Scene".
Come si nota anche dalla figura 3.8, la galleria è organizzata in diversi modi, tutti pensati per facilitare la scelta dei modelli 3D desiderati per creare la propria storia: per gerarchie di classi, secondo dei temi o regioni, secondo dei gruppi o categorie; inoltre vi è una parte per l'inserimento di figure geometriche, testo 3D ed immagini 2D oppure in alternativa è possibile cercare gli elementi per nome.

Figura 3.8: Organizzazione della galleria e scelta di un modello 3D

Come ulteriore esempio delle scelta organizzativa delle classi, si vede per esempio dalla figura 3.8 che i vari personaggi che appartengono alla classe “Biped”, sebbene abbiano tutti dimensioni, colori ed altri particolari diversi tra loro, presentano comunque delle caratteristiche comuni in quanto camminano tutti su due gambe ed avranno un modo simile di muoversi. E questa classificazione, come già sottolineato in precedenza, facilita anche la costruzione delle varie classi e l’implementazione dei metodi in esse contenuti.

3.3 Creazione di una scena

Dopo aver analizzato le varie componenti di base dell’interfaccia di Alice, è importante capire come chi utilizza questo ambiente di programmazione, possa effettivamente creare una propria scena o storia.
Per prima cosa vediamo come è possibile creare un oggetto ed inserirlo nella scena, mentre successivamente vedremo come modificarlo. Supponiamo di voler inserire un oggetto comune preso dalla galleria divisa per gerarchia di classi: per esempio scegliamo di voler inserire un “alieno”. Per farlo è sufficiente cliccare sulla relativa figura e, tenendo premuto il tasto sinistro del mouse, trascinarlo nella posizione della scena desiderata, evidenziata da un contenitore rettangolare come si vede in figura 3.9 (cliccando sopra all’oggetto una sola volta, senza tenere premuto e trascinarlo, esso verrà inserito di default al centro della scena).

Figura 3.9: Aggiungere un oggetto trascinandolo nella scena

In entrambi i casi, prima che l'oggetto venga inserito, compare una finestra in cui si può inserire il nome dell'oggetto; si può anche lasciare il nome dato di default, che, nel caso di due o più elementi della stessa classe particolare, presenta un numero incrementato automaticamente che serve come identificatore degli oggetti: questo fa già capire allo studente che in un linguaggio di programmazione orientato agli oggetti di norma non si possono avere oggetti con lo stesso nome. Infatti Alice non permette di scegliere nomi uguali anche per oggetti di tipi diversi ed inoltre dichiara come non validi i nomi che presentano spazi, seguendo appunto le regole sintattiche di Java.
Non proibisce invece di definire nomi con la prima lettera maiuscola, poiché non è proibito neanche in Java sebbene sia consuetudine evitarlo, mentre invece è in linea con la sintassi di Java proibendo di definire un nome avente un numero o un carattere che non sia "underscore" come primo carattere.
Naturalmente, è possibile inserire più oggetti della stessa classe e più oggetti di classi diverse e con ciò lo studente si rende conto dell'indipendenza degli oggetti gli uni

dagli altri, i quali possono “convivere” a meno di poter essere identificati univocamente. Un po’ più complesso è l’inserimento di una persona (adulto o bambino) poiché presenta, dopo averne selezionato la posizione e prima della scelta del nome, una serie di caratteristiche da poter impostare. Queste caratteristiche che vengono scelte, non sono altro che i parametri del costruttore che andrà a creare la nuova persona. Sebbene uno studente alle prime armi possa non capire subito questa corrispondenza, noterà comunque la presenza delle caratteristiche da lui scelte nella finestra relativa al nome nella voce “initializer”, intuendo almeno la presenza di qualche relazione tra le due parti.
Diverso è l’inserimento di una figura geometrica (figura 3.10) dove nella finestra relativa al nome vi è la possibilità di impostare valori relativi alle dimensioni, alla trasparenza o al colore della figura. Simile a questo è l’inserimento di un testo 3D che però presenta, invece della dimensione, la possibilità di inserire il testo desiderato. Come permesso anche in Java, trattandosi di una stringa, è possibile inserire anche una stringa vuota, oppure un testo non corrispondente al nome dato al testo 3D (perfettamente lecito, ma può generare confusione e scarsa leggibilità). Per quanto riguarda invece l’inserimento di un’immagine 2D (parte evidenziata in figura 3.10), nel primo campo che compare nella finestra bisogna impostare il nome, nel secondo si può o scegliere un colore oppure importare un’immagine 2D, nel terzo o scegliere un colore come sfondo oppure importare anche in questo caso un’immagine 2D contenente lo sfondo desiderato. Una volta importata un’immagine 2D, essa è mostrata anche nel menù “Resource Manager”. Se invece si vuole creare un asse, non è necessario impostare alcuna preferenza oltre al nome, ma l’asse creato può essere usato per l’orientazione degli oggetti 3D. In un mondo virtuale come quello di Alice può essere molto utile conoscere l’orientazione degli oggetti creati, e ciò è possibile creando un asse ed utilizzando il metodo “moveAndOrientTo” per fare in modo che l’asse acquisti l’orientazione dell’oggetto.

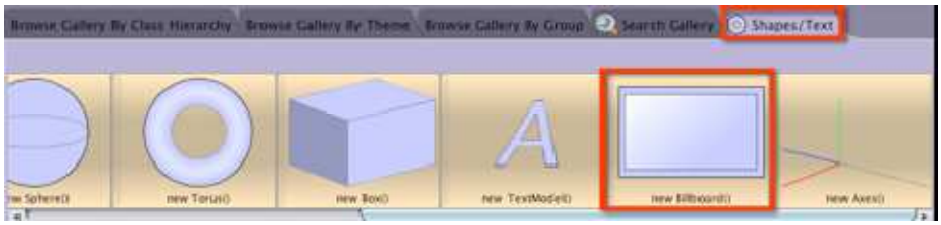


Figura 3.10: Forme geometriche, testo 3D, immagine 2D ed asse

Una volta capito come inserire correttamente gli oggetti nella scena ed impostare le loro caratteristiche di base, è necessario vedere come si impostano o modificano le altre proprietà.

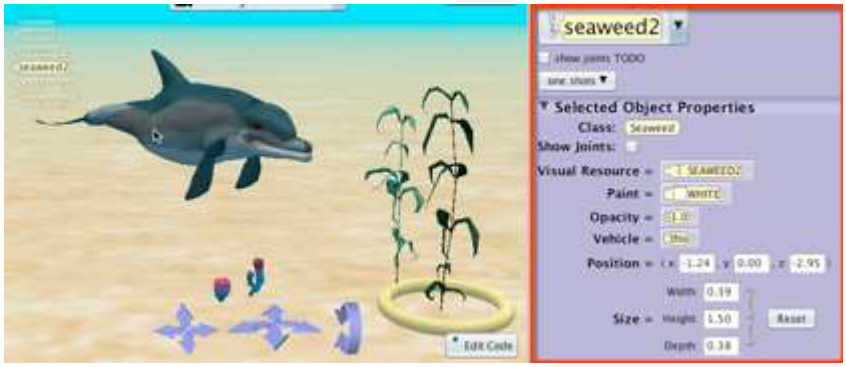


Figura 3.11: Gli oggetti e le loro proprietà

Come mostrato in figura 3.11 (dove viene mostrato il Scene Setup Panel) dopo aver creato un oggetto (ed aver già impostato alcune proprietà se richiesto), lo si può selezionare cliccandoci sopra con il mouse, e nella parte destra dello Scene Setup Panel comparirà un elenco con tutte le proprietà (anche quelle già impostate), permettendone così la modifica. Quando viene selezionato, l'oggetto viene evidenziato con un anello circolare che sta attorno alla sua base (opzione "Default") e che da la possibilità all'utente di girarlo verso destra o sinistra (muovendo l'anello con il mouse) oppure di spostarlo in qualsiasi posizione nella scena trascinandolo (l'oggetto, non l'anello) con il mouse. Nella parte destra del Scene Setup Panel, sopra al nome, vi sono alcune opzioni quali: "Default" che si ha in presenza del solo anello

circolare mostrato in figura 3.11; “Rotation”, che serve per ruotare l’oggetto in tutti i modi possibili attraverso i tre anelli (ma non è possibile trascinare l’oggetto dove si vuole come con l’opzione “Default”) e in alcuni casi permette una migliore illuminazione dell’oggetto stesso a causa del modo in cui viene colpito dalla “luce virtuale”; “Translation”, che serve per spostare l’oggetto, senza farlo girare o ruotare, in tutte e sei le direzioni, cioè verso destra o sinistra, in avanti o indietro ed in alto o in basso (ovviamente l’orientazione e l’angolo non vengono modificati);
“Resize”, che serve per ingrandire o rimpicciolire l’oggetto, mantenendo però inalterate le proporzioni (eccezione fatta per le figure geometriche, per le quali è possibile anche modificare le varie dimensioni singolarmente senza mantenere le stesse proporzioni di partenza). Vediamo quali proprietà dell’oggetto si possono modificare. Il nome, cliccando sul pulsante “one shots” e selezionando l’opzione “Rename” oppure cliccando sull’oggetto selezionato col tasto destro del mouse e selezionando la stessa opzione. Il colore, che in sostanza è una sorta di “copertura” che ha la funzione di modificare il colore normale di un oggetto; il colore di default di questo “rivestimento” è il bianco, e ciò indica che il colore di default dell’oggetto non viene modificato: più scuro è il colore, meno dettagli si riescono a distinguere dell’oggetto originale; si può o selezionare un colore tra quelli presenti o crearne uno personalizzato. L’opacità, che indica quanto l’oggetto è trasparente: di default è impostato al valore 1.0 che è il valore massimo ed indica nessuna trasparenza, ma si possono attribuire valori fino allo 0.0 che indica massima trasparenza e rende l’oggetto invisibile. È inoltre possibile modificare l’opzione “Vehicle”: essa lega il moto di un oggetto al moto di un altro oggetto sulla scena, permettendo al primo di essi (quello “veicolato” al movimento del secondo) di muoversi esattamente come il secondo, cioè nella stessa direzione, con lo stesso angolo di rotazione, con la stessa durata e per la stessa distanza. Un esempio potrebbe essere un uomo su un mezzo di trasporto: per rendere la scena realistica è necessario che l’uomo si muova esattamente come il mezzo e con il mezzo, restando sostanzialmente fermo rispetto ad esso. Bisogna notare però che mentre il primo oggetto deve necessariamente

seguire i movimenti del secondo, il secondo non è tenuto a seguire quelli del primo se tale opzione non è impostata. È sufficiente cliccare quindi sul pulsante relativo a "Vehicle" per scegliere l'oggetto desiderato (di default è impostato a "this", cioè non segue i movimenti di nessun altro oggetto, ma si muove come la scena). Oltre ai pulsanti visti sopra, è possibile modificare la posizione di un oggetto in un altro modo e con maggior precisione: come si vede in figura 3.11, sotto a "Vehicle", vi è l'opzione "Position" che permette di impostare la posizione di ogni oggetto, relativa al centro della scena, cambiando le sue coordinate. In questo modo però non verranno cambiati l'angolo e orientazione dell'oggetto ma solo la posizione relativa agli assi. È da notare che in Alice l'orientazione di un oggetto è molto importante poiché, quando vengono dati dei comandi di movimento ad un oggetto attraverso dei metodi, esso li esegue seguendo la sua orientazione: l'oggetto possiede dei suoi assi virtuali (evidenziati anche nel momento in cui viene spostato tramite lo Scene Setup Panel), che sono il suo punto di riferimento per eseguire le operazioni richieste. Per esempio se gli si ordina di spostarsi a sinistra esso non si sposterà alla sinistra dell'osservatore della scena (o rispetto alla telecamera) ma alla sua sinistra e rispetto al centro e all'orientazione dei sui assi virtuali. Come per la posizione, anche la dimensione, oltre a poter essere modificata come detto in precedenza, può essere impostata con più precisione inserendo i parametri "width", "height" e "depth": basta modificarne uno di questi e premere il tasto "INVIO" perché l'immagine venga ridimensionata e vengano calcolati automaticamente gli altri due parametri rispettando le proporzioni (questo vale anche per le figure geometriche, quindi per modificare le dimensioni separatamente è necessario usare lo strumento "Resize"). Cliccando sul pulsante "Reset", l'oggetto ritorna immediatamente alla dimensione originale. Sono presenti anche diverse opzioni per creare degli effetti speciali in una scena quali colore dell'atmosfera, luminosità e nebbia. Come si vede in figura 3.12 bisogna per prima cosa selezionare come oggetto "this", cioè la scena nel menù di destra, e poi modificare in maniera appropriata i quattro valori relativi a questi effetti.

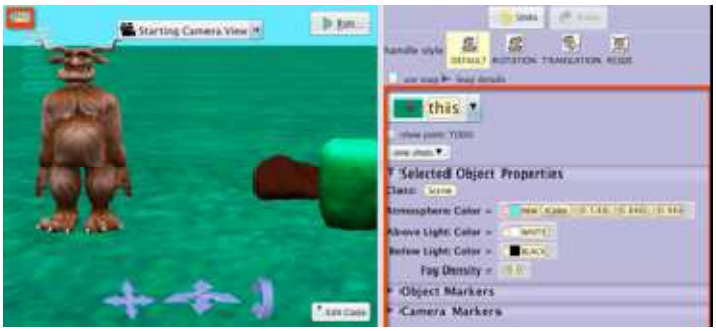


Figura 3.12: Effetti speciali per la scena

I valori mostrati in figura 3.12 sono quelli di default, cioè per i quali non viene applicato nessun effetto. “Atmosphere Color” determina il colore del cielo ed è possibile selezionarne uno nell’elenco oppure crearne uno personalizzato. “Above Light Color” indica la luminosità del cielo, ed il valore bianco di default rappresenta la luminosità del cielo in una giornata serena col sole splendente. “Below Light Color” serve per impostare un’eventuale luminosità proveniente dal basso (di default il colore nero indica che non c’è luminosità). “Fog Density” serve per creare un effetto nebbia, in cui sono più visibili gli oggetti in primo piano e meno quelli sullo sfondo (0.0 è il valore di default ed indica che non c’è nebbia, 1.0 invece indica massimo grado di nebbia e la visibilità è praticamente nulla).

Un altro elemento importante è la telecamera, ed è utile capire come modificarne la posizione in modo da creare la visuale adatta e desiderata per la scena. In Alice vi sono due metodi principali per cambiare la posizione della telecamera: i comandi di navigazione della telecamera ed i punti di vista predefiniti della telecamera (“markers”). Nella scena è presente un’unica telecamera che è possibile muovere in varie direzioni poiché nella costruzione di una scena questa è un’operazione abbastanza comune. Lo si può fare al meglio nello Scene Setup Panel (oppure anche nella sezione Code Editor) tramite comandi di navigazione costituiti da varie frecce direzionali (figura 3.12, in basso a sinistra) oppure tramite il mouse, i cui due pulsanti e la rotella corrispondono appunto a queste frecce. Si può quindi muovere la

telecamera a destra, sinistra, verso l'alto, verso il basso, avanti o indietro (zoom) oppure la si può ruotare avanti, indietro, verso destra o sinistra. Si può quindi modificare non solo la posizione ma anche l'angolo e l'orientazione in modo da ottenere la ripresa desiderata. Si può anche creare, nella parte destra dello Scene Setup Panel (figura 3.13), un "camera marker", che è un oggetto che contiene i valori della telecamera, cioè la visuale attuale, al momento della creazione del marker. Una volta creato, tale oggetto mantiene salvati questi valori ed è possibile muovere la telecamera a proprio piacimento senza rischiare di perderli. Cambiando la visuale (per esempio spostando indietro la telecamera) dopo aver creato un marker , si noterà che tale oggetto (a cui si può dare un colore che non deve essere univoco per ogni marker, anche se ciò è fortemente consigliato per evitare confusione) compare sulla scena ed è come un punto fisso da cui la si può riprendere. Premendo sul pulsante "Run" però si noterà che al momento dell'esecuzione, anche se il marker è inquadrato, esso non viene visualizzato. Cliccando su un marker (nella parte a destra del Scene Setup Panel e non nella scena) si riporta la telecamera alla posizione indicata dal marker. Se vi sono più marker, cliccando su uno di essi sulla scena, si vedrà una linea corrispondente alla direzione della visuale e gli altri marker diventeranno trasparenti. Essendo essi simili a degli oggetti, è possibile spostarli anche utilizzando il mouse. Nella parte destra del Scene Setup Panel , oltre al pulsante per creare un marker, vi sono pulsanti per cambiare marker o passare da uno all'altro cambiando così la visuale (figura 3.13).
Dalla figura 3.13 si nota inoltre che è possibile impostare anche il "Vehicle" trai parametri della telecamera, potendo quindi fare in modo che segua i movimenti di qualche oggetto. È anche possibile cambiare la posizione della telecamera tramite l'opzione "Position", cioè modificando le sue coordinate sugli assi (ciò però non cambia l'orientazione e l'angolo). Dal punto di vista pratico è consigliabile salvare la posizione iniziale della telecamera in un camera marker e successivamente spostarla a proprio piacimento.

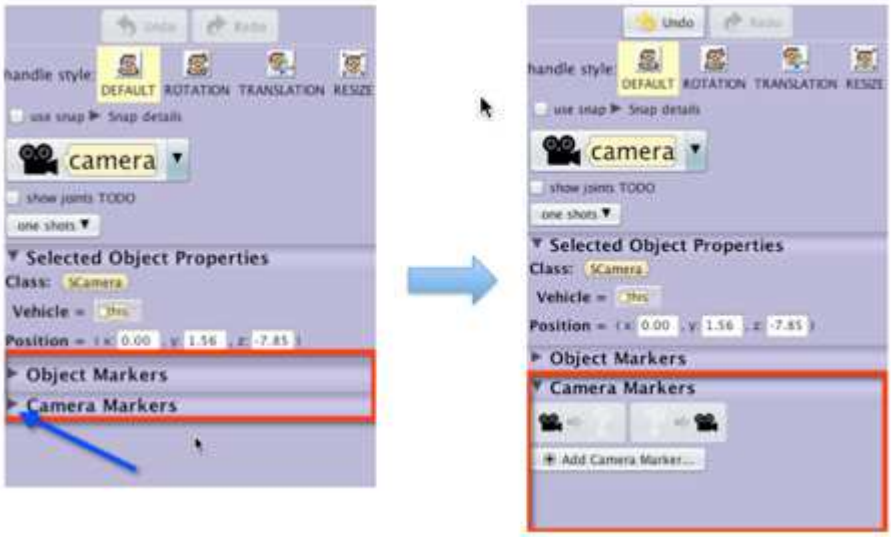


Figura 3.13: Impostazioni per la telecamera

Oltre a quelli per la telecamera, è possibile creare dei marker anche per gli altri oggetti: essi ne salvano la posizione e l'orientazione (relativa al momento della creazione del marker) ed ognuno di essi è costituito da un insieme di assi posto alla base dell'oggetto. È quindi possibile ripristinare una posizione voluta dopo aver fatto dei cambiamenti, con modalità e pulsanti del tutto simili a quelli relativi ai camera markers (in basso a destra nel Scene Setup Panel). Come per la telecamera, vi è anche l'apposito pulsante per riposizionare il marker, per esempio nella posizione di un oggetto.

In Alice vi è anche la possibilità di posizionare delle sottoparti di un oggetto nella scena. I modelli 3D di Alice sono costituiti da un sistema scheletrico interno che consiste in vari collegamenti. Per esempio delle parti del corpo come testa ed arti sono connesse l'una con l'altra e con il corpo attraverso questi collegamenti. Pertanto, una sottoparte di un oggetto viene posizionata ruotando i collegamenti del sistema scheletrico. Per visualizzare questi collegamenti è necessario per prima cosa abilitare, nella parte destra del Scene Setup Panel (figura 3.14), l'opzione "Show Joints", e poi modificare l'opacità dell'oggetto inserendo un valore basso come per

esempio 0.5 (ottenendo un effetto “radiografia”). Da notare che la posizione di ogni collegamento è contrassegnata con dei piccoli assi contraddistinti da colori diversi per ogni direzione: bianco in avanti, rosso a destra e verde in alto. Gli assi hanno lo scopo di mostrare la posizione e l’orientazione dei collegamenti, che per la maggior parte dei quali è coerente con il funzionamento di una sottoparte collegata. Inoltre alcune sottoparti, data la loro complessità o grandezza, presentano più di un collegamento, per renderne più realistico e dettagliato il movimento ed avere maggior flessibilità di animazione (i collegamenti devono avere la stessa orientazione).

Figura 3.14: Collegamenti per le sottoparti

Le sottoparti di un oggetto possono essere posizionate nella scena selezionando (come in figura 3.15) il collegamento appropriato nel menù “Object Parts” (parte destra del Scene Editor Panel).

Quando un collegamento viene selezionato, Alice seleziona automaticamente l’opzione “Rotation” caratterizzata dai tre anelli. È infatti possibile ruotare o cambiare l’orientazione di una parte di un oggetto ma non è possibile cambiarne la posizione, staccandola quindi dall’oggetto.

Figura 3.15: Selezionare una sottoparte di un oggetto

Un altro obiettivo da raggiungere è quello di capire come posizionare correttamente un oggetto nella locazione relativa ad un altro: per esempio può essere utile posizionare un utensile al centro di un tavolo. Per il posizionamento relativo di questi oggetti in Alice ci si aiuta attraverso i punti di vista multipli della telecamera; non basta trascinare l'oggetto come fatto finora, dato che in questo modo sarebbe facile commettere errori: infatti noi osserviamo la scena dal punto di vista della telecamera, per cui potremmo non accorgerci se un oggetto non è perfettamente al centro di un altro. Nella parte in alto a sinistra del Scene Editor Panel vi è il menù "Camera Viewpoints", che permette di selezionare degli altri punti di vista predefiniti attraverso i quali osservare la scena (figura 3.16).

Figura 3.16: Punti di vista predefiniti della telecamera

In figura 3.16 si sta selezionando il punto di vista “Layout Scene View”, che è una sorta di visuale panoramica dall’alto (non appena lo si seleziona la visuale cambia automaticamente). Da questa visuale ci si accorge che per esempio la teiera in figura non è perfettamente al centro del vassoio come si poteva erroneamente pensare, e quindi ora la si può riposizionare correttamente, poiché questo punto di vista ha le stesse funzioni (viste in precedenza) del punto di vista di partenza della telecamera che prima era utilizzato. Vi sono altri punti di vista come la vista dall’alto (vera e propria) “TOP”, la vista laterale (la visuale è dalla parte destra rispetto al terreno) “SIDE” e quella frontale (dal punto di vista del centro del terreno) “FRONT”, le quali però, rispetto alle prime due, presentano delle limitazioni in termini di operazioni di riposizionamento che possono essere effettuate sugli oggetti della scena e sullo spostamento della telecamera stessa che inquadra la scena.

Può essere utile inoltre saper allineare due o più oggetti utilizzando l’apposita griglia messa a disposizione da Alice: è sufficiente attivare l’opzione “use snap” nella parte destra del Scene Setup Panel per visualizzare la griglia (quadrata). Successivamente con il pulsante “Snap details” si possono vedere ed eventualmente cambiare i dati relativi alla griglia come la dimensione dei quadrati. La griglia è molto efficace per l’allineamento degli oggetti, poiché non è solo un punto di riferimento visivo, ma quando si sta muovendo un oggetto trascinandolo ed esso si trova su una riga della griglia, essa viene evidenziata, permettendo quindi di allineare correttamente più oggetti sulla stessa riga.

Vi è un ulteriore modo per allineare o posizionare precisamente gli oggetti o le loro sottoparti nella scena: nella parte destra del Scene Setup Panel, oltre agli altri pulsanti, vi è quello relativo al menù “one shots” che permette di utilizzare procedure (metodi) come quelle presenti nel Code Editor per spostare gli oggetti. Le stesse procedure vi sono anche nella parte in alto a sinistra del Scene Setup Panel: è sufficiente cliccare col tasto destro del mouse sul nome dell’oggetto voluto (figura

3.17) oppure sull’oggetto all’interno della scena e poi selezionare il metodo che interessa.

Figura 3.17: Modo alternativo per aprire il menù “one shot”

Il metodo selezionato viene applicato all’oggetto una e una sola volta ed in quel preciso istante (per esempio facendo muovere un oggetto in avanti si ottiene lo stesso effetto di posizionamento che si otterrebbe muovendolo usando il mouse). Usando questo metodo piuttosto che il trascinamento col mouse, l’unica differenza è che si può ottenere una maggiore precisione nel posizionamento: con le tecniche utilizzate in precedenza non era facile posizionare un oggetto a centro di un altro e bisognava porre una certa attenzione. Invece con questa tecnica è sufficiente utilizzare il metodo “moveAndOrientTo” per ottenere lo stesso effetto in maniera più semplice e veloce senza il rischio di commettere errori. Utilizzando questo ed altri metodi è anche possibile allineare correttamente gli oggetti, ricordando però che la distanza calcolata per esempio da due oggetti viene misurata tra i rispettivi centri. Sfruttando il menù “one shot”, avendo prima selezionato una sottoparte di un oggetto, è possibile cambiarne l’orientazione utilizzando dei metodi appropriati (che sono solamente cinque per il fatto che una sottoparte non può essere staccata dall’oggetto) e questa rappresenta un’ulteriore alternativa per la modifica delle posizioni delle sottoparti descritta in precedenza.

Va sottolineato inoltre che modificare una sottoparte di un oggetto ha un effetto su tutte le sottoparti ad essa associate (per esempio se si fa ruotare la spalla di un uomo di un certo angolo anche il braccio e la mano cambieranno orientamento) .

3.4 Creazione di un programma

Grazie a tutti gli utili strumenti descritti nella sezione precedente, è quindi possibile creare l'ambiente ed i personaggi desiderati per la propria storia, nonché la visualizzazione di partenza per la scena. La cosa più importante per realizzare gli scopi per cui questo software è stato ideato, è però l'animazione della scena, che si ottiene "dando vita" al mondo virtuale, ovvero scrivendo un programma.
È possibile scrivere un programma sia utilizzando la sintassi di Alice che quella di Java. Come già detto è sufficiente trascinare i metodi nell'editor con le combinazioni appropriate per creare un programma sensato e funzionante, senza doversi preoccupare di commettere errori di sintassi.

La cosa più semplice per imparare ad utilizzare Alice e per introdurre alcuni concetti riguardanti l'editor è però la pratica, per cui risulta utile vedere come si può costruire un semplice programma contenente i costrutti principali di Alice.

Innanzitutto è necessario immaginare una possibile storia o situazione e, dopo aver costruito la scena in maniera adeguata (cosa che viene data per scontata in quanto spiegata nelle sezioni precedenti), si può iniziare a creare le varie righe di codice sfruttando l'intuitivo ambiente "drag and drop".

Immaginiamo un breve racconto per focalizzarci su alcuni elementi e costrutti. Scriveremo prima il programma utilizzando la sintassi di Alice, e successivamente quella simile a Java. La storia è molto semplice: l'ambientazione è la casa di una strega che sta preparando una pozione per evocare un drago. Una volta evocato gli fa un incantesimo in modo tale da farlo obbedire ai suoi ordini e, dopo aver festeggiato, si avvicina al drago per cavalcarlo ed andare a conquistare il mondo. A questo punto il programma diventa interattivo: spetta infatti a chi sta osservando la scena decidere se lasciarli andare liberamente oppure se far intervenire un mago per fermarli.

Figura 3.18: Scena iniziale

In figura 3.18 è rappresentata la scena di partenza costruita utilizzando le modalità descritte nelle sezioni precedenti. Si noti che il drago ed il mago, pur essendo invisibili, sono presenti sulla scena: è stato utilizzato il metodo "setOpacity" al di fuori del programma per fare in modo che non si vedano (compariranno successivamente).

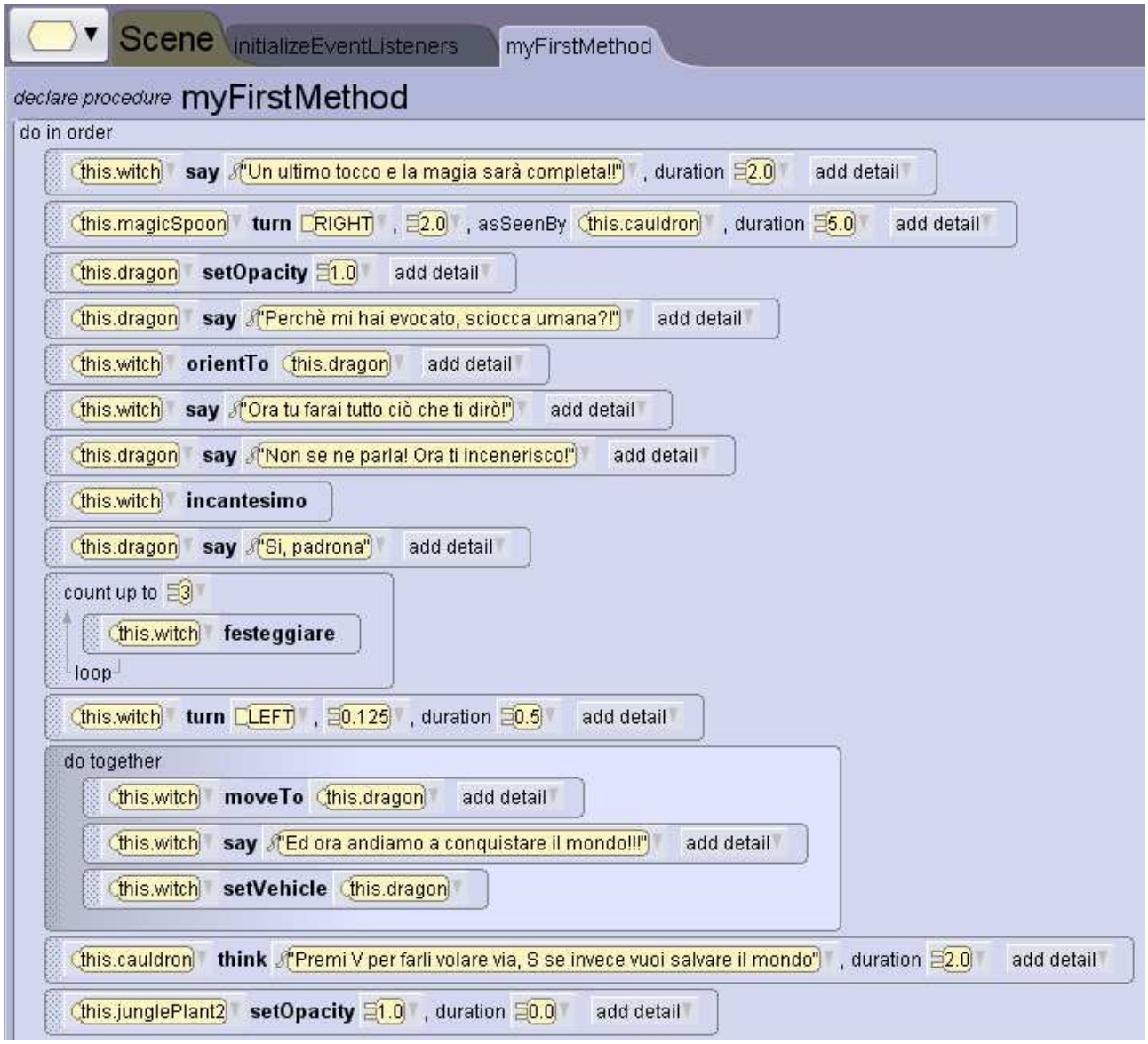


Figura 3.19: Una possibile implementazione della storia scelta

In figura 3.19 è rappresentato il codice relativo ad uno dei possibili modi con cui può essere creata la scena scelta. La sintassi utilizzata è quella di Alice. Il codice potrebbe sembrare poco, ma in realtà è solamente quello riguardante “myFirstMethod”,che unisce anche altre parti del programma sviluppate come metodi separati. Prima di procedere con l’analisi, bisogna sottolineare che lo scopo da raggiungere con un programma in Alice non è la perfezione estetica, bensì la comprensione dei concetti basilari riguardanti la programmazione orientata agli oggetti. Per cui eseguendo il programma si nota subito che dal punto di vista estetico si potrebbero apportare modifiche atte a migliorare l’espressività, la gestualità ed i movimenti dei personaggi,

ma questo va al di là degli scopi prefissati, almeno per quanto riguarda Alice 3 (più avanti si vedrà che invece in Storytelling Alice viene data maggiore importanza a questi dettagli e alla realizzazione di storie anche graficamente belle).
Per quanto riguarda le prime sette righe di codice, esse sono delle semplici istruzioni di facile lettura che si ottengono trascinando i metodi dal Methods Panel, scegliendo nel menù in alto l'oggetto che deve compiere il metodo e scegliendo inoltre tra procedure e funzioni. Riassumendo, nelle prime sette righe di codice la strega mescola il calderone e compare il drago che si rifiuta di obbedirle. Nell'ottava riga di codice è rappresentato un metodo non presente per quanto riguarda la strega, ma creato a parte e presente in un'altra parte del programma: per creare dei metodi personalizzati è sufficiente selezionare l'oggetto desiderato dall'apposito menù vicino al pulsante "Scene" (figura 3.19 in alto a sinistra), e scegliere la voce desiderata (si possono creare altre cose come procedure, funzioni, e proprietà come mostrato in figura 3.21). In figura 3.20 sono mostrati i due metodi creati per la strega, cioè "incantesimo" (presente nell'ottava riga di codice) e "festeggiare". Si può accedere alla sezione contenente il loro codice per modificarlo premendo il pulsante "edit".

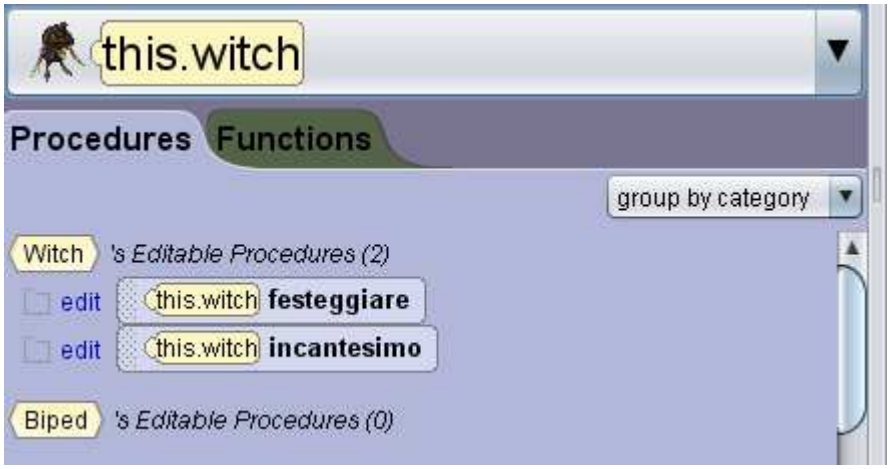


Figura 3.20: Metodi personalizzati

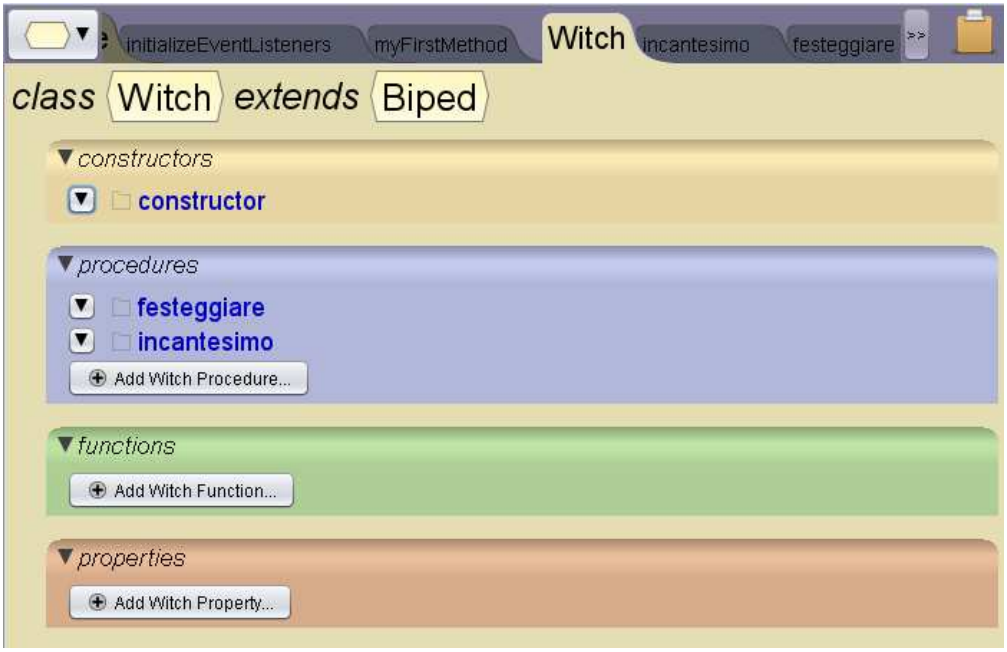


Figura 3.21: Aggiungere e modificare procedure, funzioni e proprietà

La riga “class Witch extends Biped” in figura 3.21 è un chiaro esempio di come in Alice 3 si cerchi di dare l’idea del concetto di ereditarietà: la parola “extends” è infatti presente anche in Java e serve per dire che una classe deriva da un’altra (ed è quindi una sua sottoclasse).

In figura 3.22 è riportato il codice del metodo “incantesimo” (dell’ottava riga in “myFirstMethod”). La strega pronuncia semplicemente una formula magica e muove le braccia. Le cose significative da notare sono però l’uso della struttura “do together” per eseguire le istruzioni contemporaneamente e l’uso delle sottoparti dell’oggetto in questione per ottenere il movimento delle braccia.

Figura 3.22: Codice relativo al metodo “incantesimo”

Tornando alla figura 3.19, la nona riga di codice è costituita da un metodo comune, mentre nella decima è utilizzata la struttura “count up to”, che è un ciclo grazie al quale si possono ripetere delle istruzioni quante volte si vuole. In questo caso il ciclo ripete per tre volte il metodo “festeggiare” relativo alla strega, il cui codice è riportato in figura 3.23 ed in cui non viene fatto nulla di particolare: la strega semplicemente salta ed esulta. Sarebbe stato anche possibile ottenere lo stesso effetto del ciclo copiando per tre volte l’istruzione “this.witch incantesimo” in “myFirstMethod” oppure scrivere tre volte di seguito le stesse istruzioni del metodo “festeggiare”, ma sarebbe stato lungo e noioso.

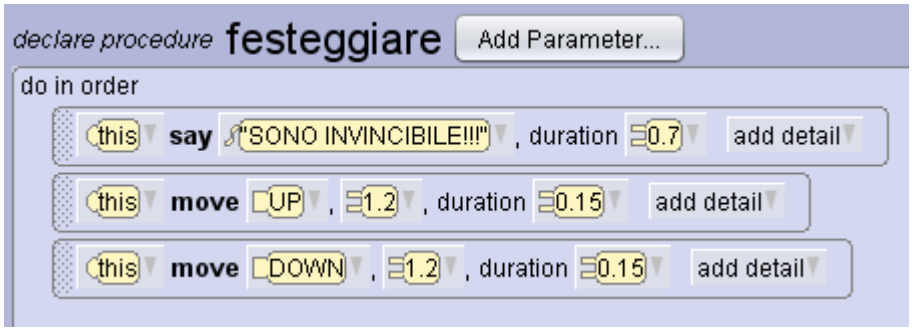


Figura 3.23: Codice relativo al metodo “festeggiare”

L’undicesima riga di codice in “myFirstMethod” rappresenta un classico metodo di movimento mentre le successive tre righe sono contenute in una struttura “do together” per cui sono eseguite contemporaneamente. Da notare l’uso di “setVehicle” per fare in modo che la strega si muova come il drago, cioè voli insieme a lui e non resti ferma. La penultima riga di codice serve a far spuntare una nuvoletta dal calderone che dice a chi sta osservando la scena di premere “V” per farli volare via oppure “S” per far intervenire il mago e salvare il mondo, mentre l’utilità dell’ultima riga verrà spiegata successivamente. Sarebbe stato possibile avvertire l’utente anche in altri modi, come inserire un testo 3D in quel preciso momento e poi farlo scomparire oppure avvisarlo utilizzando l’audio, ma non sarebbe servito a spiegare ulteriori meccanismi con cui è possibile scrivere i programmi. A questo punto il programma resta in attesa che l’utente decida cosa scegliere e rimane bloccato finché egli non preme una di quelle due lettere o termina il programma in un altro modo. Questo serve per introdurre un’altra parte molto importante relativa ai programmi, e cioè la gestione degli eventi. Di default, quando viene fatto partire il programma, Alice esegue automaticamente “myFirstMethod”, ma è possibile modificare tale opzione: per farlo è sufficiente scegliere il metodo “initializeEventListeners” relativo alla classe “Scene” dal pannello dei metodi (come fatto prima per i metodi della strega), oppure dalla gerarchia di classi nell’editor (di cui una parte è mostrata in figura 3.24).

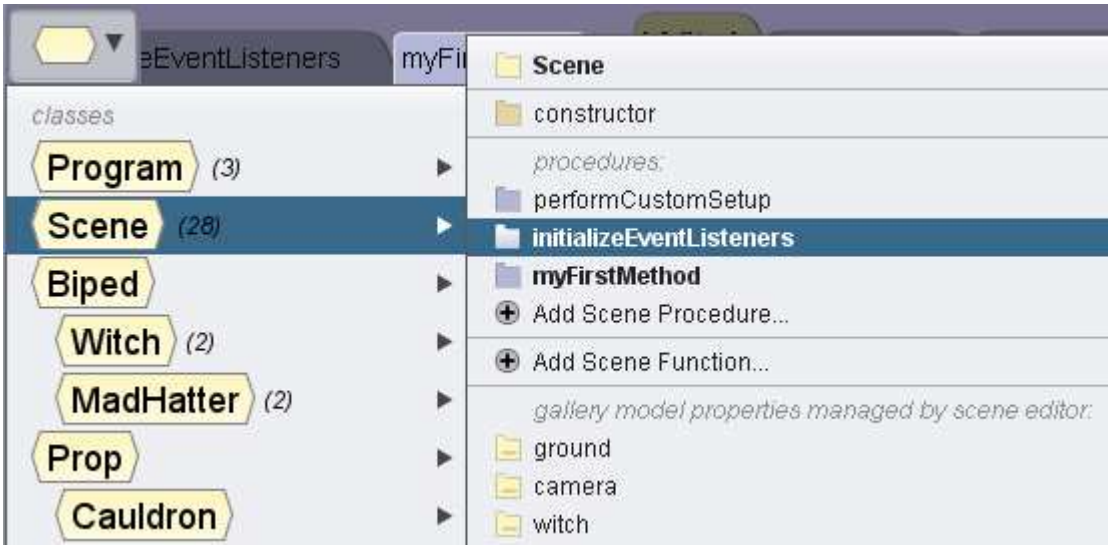


Figura 3.24: Metodo alternativo per aprire i metodi e le classi

Il primo evento che si vede aprendo tale classe è “addSceneActivationListener” (figura 3.25), che permette di eseguire “myFirstMethod” all’avvio del programma. È possibile aggiungere molteplici eventi di questo tipo nel programma per avviare contemporaneamente altri metodi e parti di programma create, facendo però molta attenzione, poiché è facile sbagliare e commettere errori logici.

Figura 3.25: Uso dell’evento “addSceneActivationListener”

L’altro evento presente nel programma, con il relativo codice, è riportato in figura 3.26. Esso serve per gestire ciò che succede quando viene premuto un pulsante della tastiera dall’utente, e crea quindi l’interattività del programma, rendendolo paragonabile ad un semplicissimo videogioco.

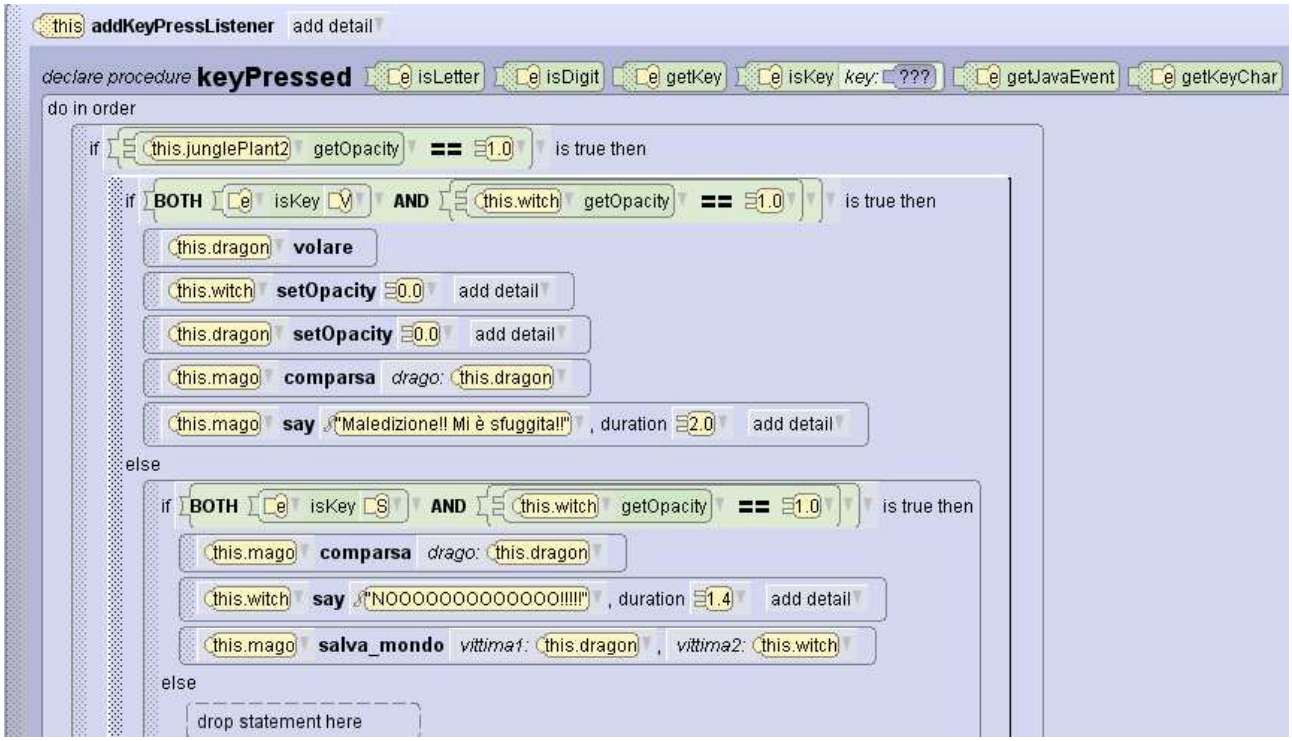


Figura 3.26: Evento “addKeyPressListener” presente nel programma

Per quanto riguarda la gestione degli eventi realizzata da Alice, bisogna tener presente una cosa fondamentale: è come se il programma eseguisse un ciclo infinito, rimanendo costantemente in attesa che si verifichi un evento, che in questo caso è che l’utente prema un pulsante. Qualsiasi pulsante venga premuto ed in qualsiasi momento, il programma inizia ad eseguirne il codice per poi fermarsi ed attendere nuovamente che venga premuto un altro pulsante. Per questo è bene inserire le istruzioni in strutture condizionali, scegliendo i pulsanti per i quali eseguire il codice, se si vuole evitare che il codice venga eseguito in qualsiasi momento e premendo qualsiasi pulsante. Dalla figura 3.26 si nota che tutto il codice relativo all’evento è contenuto in una struttura condizionale (“if”) con una condizione particolare: “if this.junglePlant2 getOpacity == 1.0 is true then”. Apparentemente questa istruzione non ha molto senso e quindi per comprenderne il significato e le motivazioni per cui è stata utilizzata è necessario sottolineare una cosa: trattandosi di codice ripetuto ogni volta che viene premuto un pulsante, definirvi all’interno dei metodi relativi alla scena oppure delle variabili booleane per gestirne l’esecuzione è decisamente difficile

(se non impossibile). Infatti inserendo all'inizio dell'evento un'istruzione relativa ad una variabile (oppure un metodo), senza una struttura condizionale, schiacciando un tasto durante l'esecuzione del programma, si avrebbe anche l'immediato cambiamento di stato della variabile (o l'esecuzione del metodo). Questo vanificherebbe ogni eventuale tentativo di utilizzare tale variabile (o metodo) per gestire i momenti in cui l'evento è abilitato (cioè in cui può essere eseguita quella parte di codice in esso contenuta). In particolare in questo caso si vuole evitare che la strega ed il drago possano volare via (oppure che il mago possa salvare il mondo, essendo le parti di codice da eseguire in alternativa l'una all'altra), prima che l'esecuzione del programma sia arrivata all'ultima riga del codice presente in "myFirstMethod". Per fare ciò diventa utile l'istruzione condizionale nella prima riga di figura 3.26. Infatti stabilisce che la parte di codice in essa contenuta debba essere eseguita solo quando l'opacità dell'oggetto "junglePlant2" ha il valore 1.0: quindi, anche se vengono premuti continuamente dei tasti, se la condizione non è verificata, la parte di codice non viene eseguita. L'oggetto "junglePlant2" è un oggetto che non ha una particolare funzione nel programma, ma è semplicemente un oggetto di sfondo che è stato appositamente inserito in una posizione della scena non inquadrata dalla telecamera, in modo tale che il suo stato possa essere cambiato senza che, durante l'esecuzione, si notino i cambiamenti. Ed è proprio il cambiamento di stato relativo all'opacità ad essere stato arbitrariamente scelto per abilitare l'esecuzione del codice: infatti il valore iniziale dell'opacità è stato posto a 0.0, cioè differente da 1.0. Quindi finché esso non viene cambiato, il codice non può essere eseguito. E tale cambiamento avviene proprio nell'ultima riga di codice di "myFirstMethod" (in cui l'opacità viene posta a 1.0), con durata 0.0 in modo che non si debba attendere ulteriore tempo da quando vengono date le informazioni all'utente per la scelta. Oltre alla condizione relativa al momento in cui abilitare il codice, si vuole che esso non possa essere eseguito più di una volta: una volta effettuata la scelta, deve essere eseguita la parte di codice relativa ad essa, una ed una sola volta, ed anche la parte di codice non scelta non può più essere eseguita. Per fare ciò si poteva utilizzare ancora

il trucco dell'oggetto fittizio, oppure si potevano sfruttare degli oggetti già presenti, in quanto in questa particolare situazione era più sensato dal punto di vista logico. Infatti, se la strega ed il drago riuscissero a volare via, non sarebbero più visibili nella scena, quindi utilizzare su di loro il valore dell'opacità come condizione non causerebbe un malfunzionamento del programma. Nell'altro caso invece, come vedremo, per riuscire a salvare il mondo, il mago li deve sconfiggere, ed essi vengono fatti sparire dalla scena non appena questo succede. Quindi nella seconda riga di codice in figura 3.26, la parte relativa alla vittoria della strega viene eseguita solo se si verificano contemporaneamente le due condizioni: viene premuta la lettera "V" e l'opacità della strega è 1.0 (la strega è ancora presente, cioè è la prima volta che quella parte di codice viene eseguita). Si sarebbe potuto utilizzare allo stesso modo anche il drago, oppure entrambi come ulteriore controllo. La terza istruzione è relativa al semplicissimo metodo "volare" realizzato per il drago, il cui banale codice è rappresentato in figura 3.27. Si sarebbe potuto creare un'animazione più complessa ed esteticamente migliore muovendo adeguatamente svariate sottoparti del drago, ma come già sottolineato in precedenza, questo risulta abbastanza superfluo.

Figura 3.27: Codice relativo al metodo "volare"

La quarta e la quinta riga servono per cambiare il valore dell'opacità della strega e del drago e portarli a 0.0, per disabilitare un'eventuale ripetizione del codice. Più significativa è invece la sesta riga: "this.mago comparsa drago: this.drago". In questa riga non solo vi è un metodo personalizzato definito appositamente per il mago, ma il metodo richiede anche che gli venga passato come parametro un oggetto della classe "Dragon". Quindi si capisce che in Alice possono essere creati dall'utente anche dei

metodi che richiedono dei parametri. In figura 3.28 è riportato il codice del metodo “comparsa”, dal quale si riesce a comprendere come possano essere creati dei metodi richiedenti dei parametri.

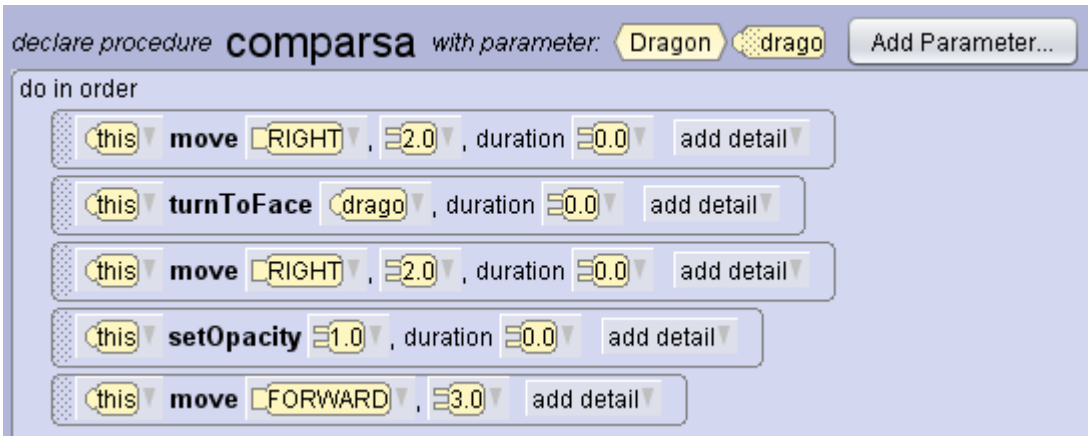


Figura 3.28: Codice relativo al metodo “comparsa”

Utilizzando il pulsante “Add Parameter” , è stato creato un parametro oggetto generico, lo si è scelto relativo ad un’istanza della classe “Dragon” e lo si è chiamato drago. Esso non è un vero e proprio parametro reale, ma un parametro oggetto generico e fittizio, che andrà sostituito con un’istanza, presente sulla scena, della classe scelta, in fase di invocazione del metodo. In questo caso è banalmente utilizzato per orientare adeguatamente il mago nella direzione di quel parametro oggetto. Uno dei motivi per cui risulta molto utile dividere il programma in metodi come questo, oltre alla semplificazione che ciò comporta, è la possibilità di poter riutilizzare più volte lo stesso metodo senza dover riscrivere più volte lo stesso codice. Tale metodo infatti verrà utilizzato anche in un’altra parte di codice.
Tornando al codice di figura 3.26, la settima riga serve semplicemente per far dire una frase al mago, dopo di che l’analisi passa all’ottava riga, dove si entra nella parte “else” relativa al secondo “if”. In questa riga vi è un’altra condizione, che si comporta in maniera analoga a quella della seconda riga: la parte di codice sottostante viene eseguita se e solo se la lettera premuta è la “S” e l’opacità della strega è posta al valore 1.0. Ciò evita che tale parte di codice possa essere eseguita dopo la “vittoria

della strega” o più di una volta. La nona riga è, come già anticipato, la ripetizione del metodo “comparsa”, che utilizza sempre il drago come parametro oggetto.
La penultima riga è semplicemente un’esclamazione della strega, mentre l’ultima è relativa ad un altro metodo scritto per il mago, cioè “salva_mondo” , a cui vanno passati due parametri oggetto, il primo di classe “Dragon” ed il secondo di classe “Biped”. Il codice di tale metodo è riportato in figura 3.29.

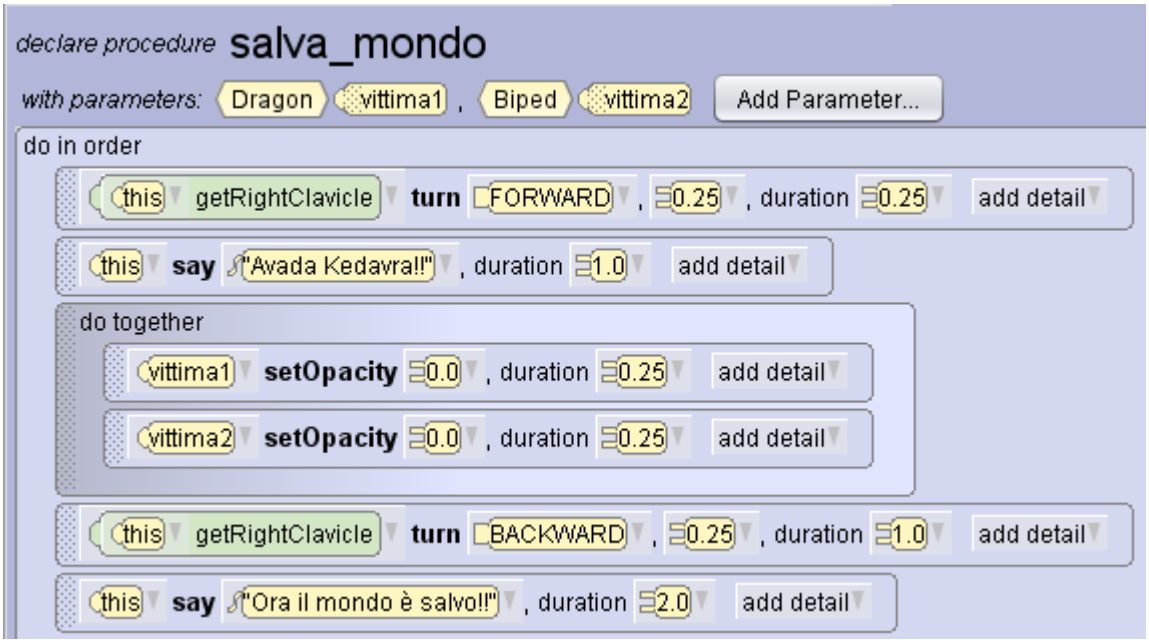


Figura 3.29: Codice relativo al metodo “salva_mondo”

Si notino l’uso dei parametri oggetto, delle sottoparti del mago ed il cambiamento contemporaneo dell’opacità dei due parametri generici (che andranno poi sostituiti con il drago e con la strega) tramite la struttura “do together”.
Con questo si conclude l’analisi dell’evento (ma ricordiamo che ce ne sono molti altri di vario tipo che si possono inserire) ed anche di questo programma creato come esempio. Per completezza e per dare la possibilità di fare un confronto, di seguito sono riportate delle figure contenenti le stesse parti di codice scritto con la sintassi simile a Java di Alice 3 (escluso il codice relativo al metodo “volare” in quanto estremamente semplice). Essa risulta meno immediata da comprendere per gli

studenti alle prime armi, però è sicuramente in linea con gli scopi che questo software si propone di realizzare.

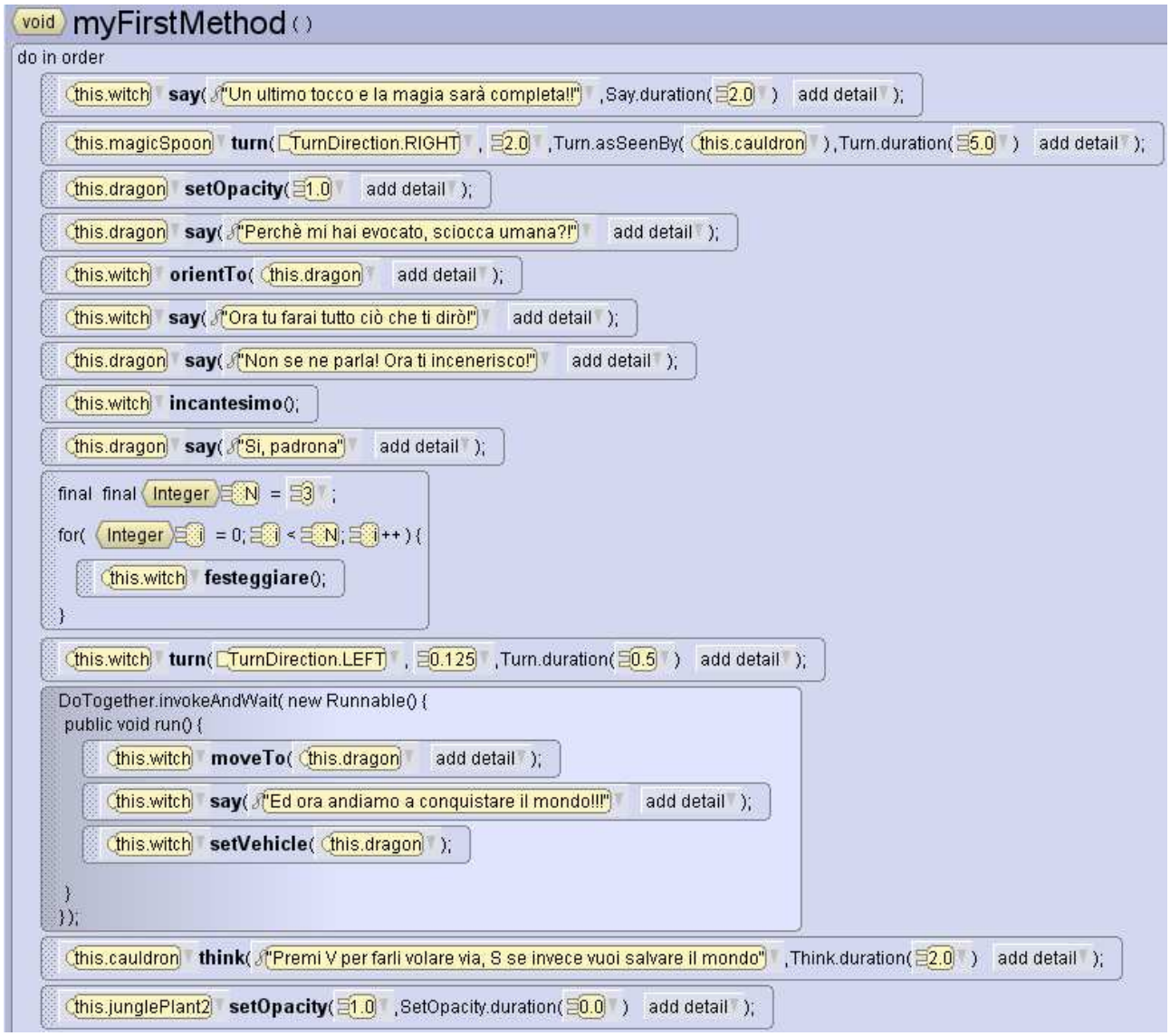


Figura 3.30: Codice “Java-like” di “myFirstMethod”

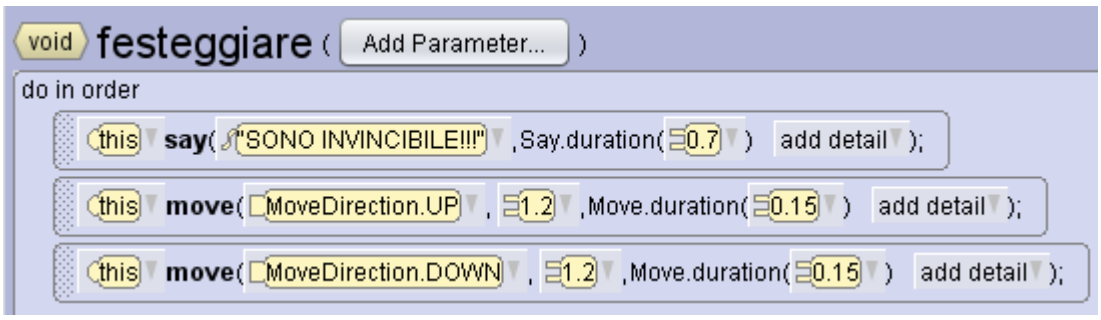


Figura 3.31: Codice “Java-like” del metodo “festeggiare”

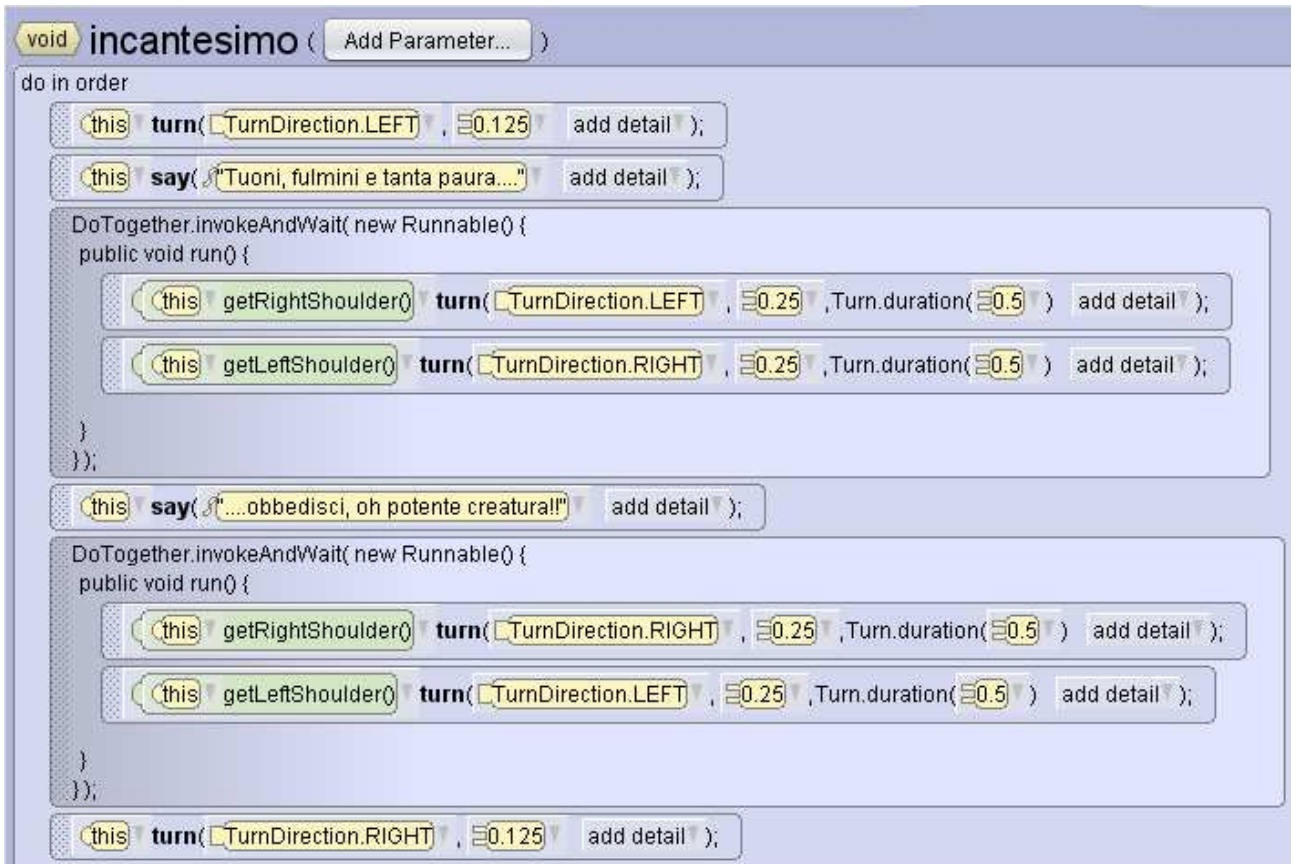


Figura 3.32: Codice “Java-like” del metodo “incantesimo”

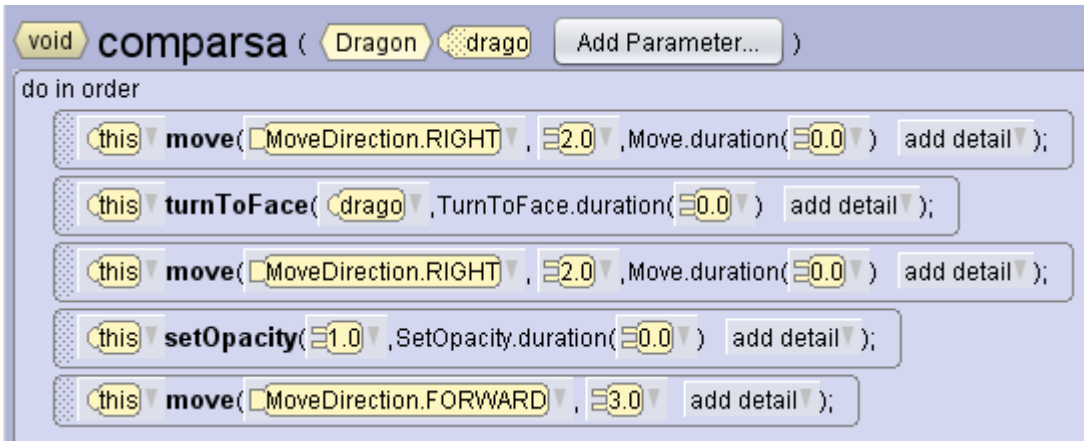


Figura 3.33: Codice “Java-like” del metodo “comparsa”

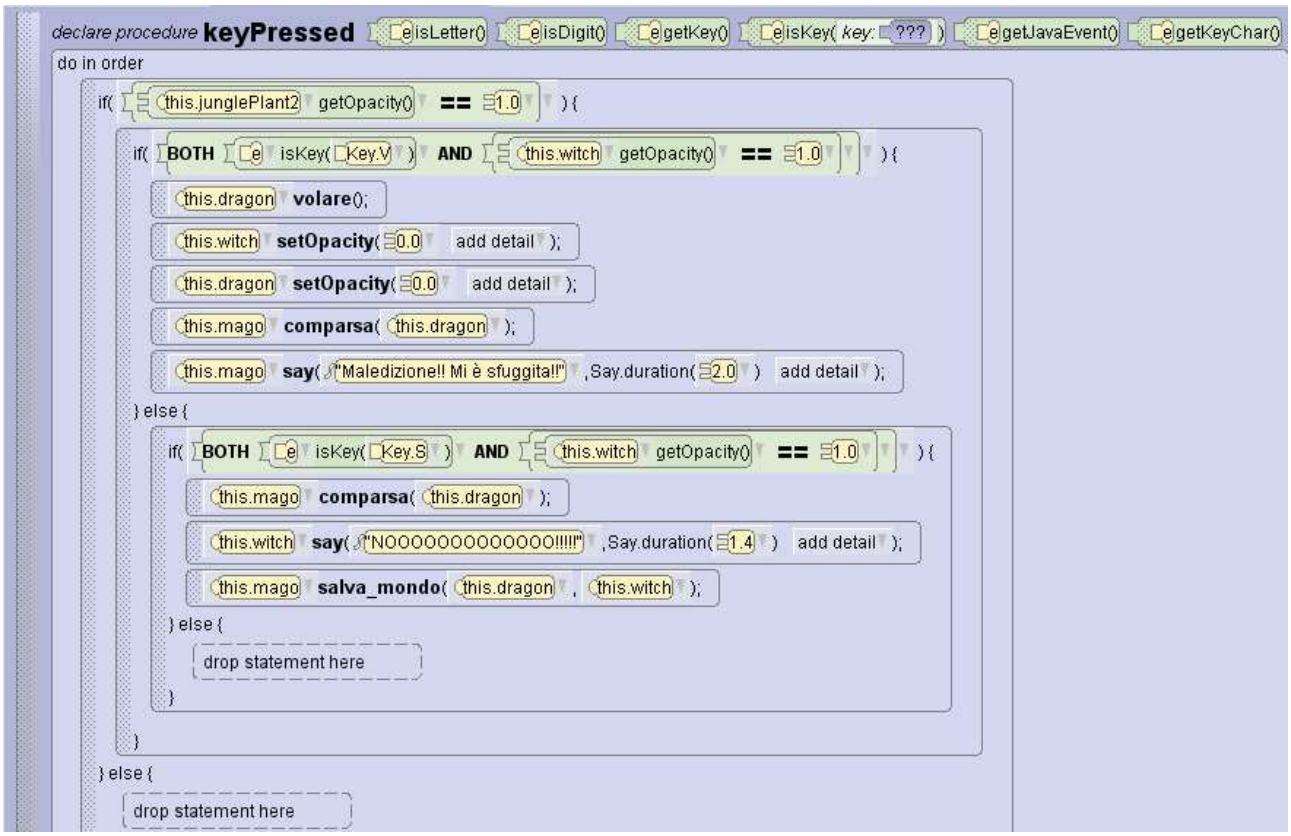


Figura 3.34: Evento “addKeyPressListener” scritto con la sintassi “Java-like”

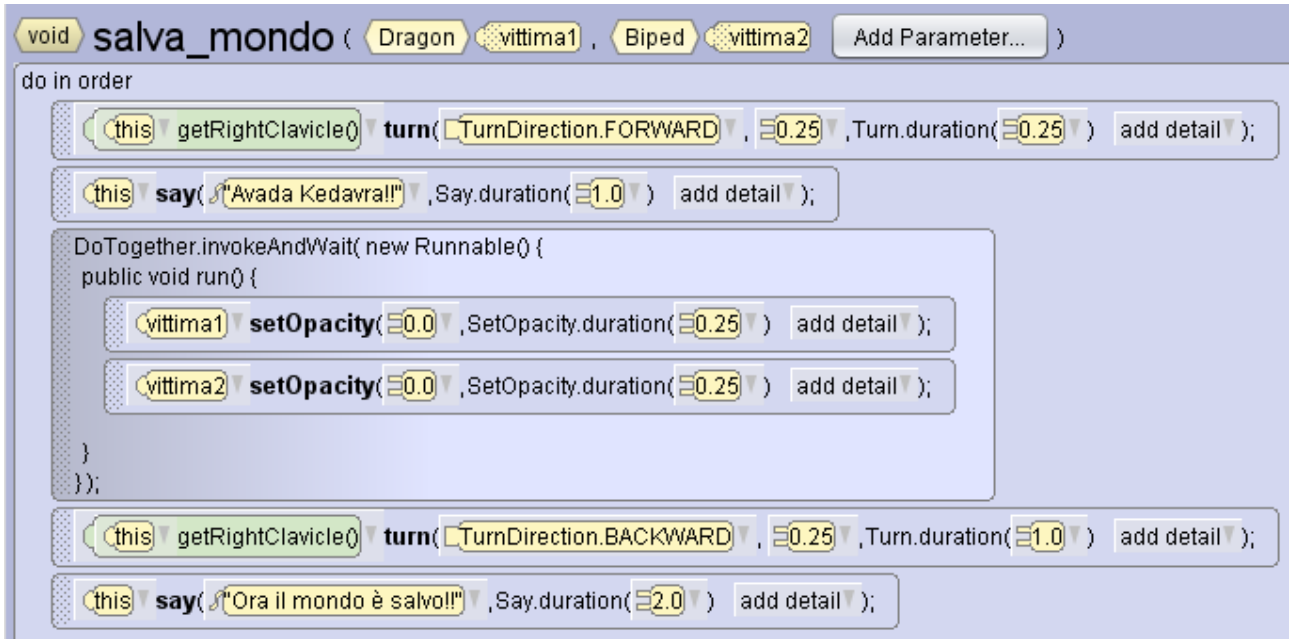


Figura 3.35: Codice “Java-like” del metodo “salva_mondo”

4 ALICE 2.X

4.1 Introduzione

In Alice 2 non si deve usare la tastiera per scrivere codice come nei classici linguaggi di programmazione ma è sufficiente l'uso del mouse grazie all'ambiente “drag and drop” che permette di trascinare le righe di codice nell’editor appropriato e successivamente vedere l’esecuzione del programma premendo il pulsante “Play” come evidenziato in figura 4.1.

Il programmatore qui diventa quasi come un coreografo o un regista che decide cosa devono fare i personaggi che egli stesso ha creato.
Lo scopo fondamentale di Alice è quello di far capire tutte le idee fondamentali coinvolte nella programmazione poiché questo faciliterà il successivo apprendimento di linguaggi di programmazione come Java, C++ o C#, dei quali sarà necessario imparare solamente le regole grammaticali e la sintassi.

Nonostante Alice non sia un vero e proprio linguaggio di programmazione, ma più che altro un modo per scrivere dello pseudocodice, esso presenta una certa “eleganza”, cioè è facilmente leggibile, e saper scrivere del codice leggibile è una delle cose che un programmatore dovrebbe essere in grado di apprendere, poiché in tal modo il codice sarebbe facilmente comprensibile anche per altri programmatori. Sempre per la stessa ragione, consente, all’interno dell’editor, l’inserimento di commenti da parte dell’utente, solitamente utilizzati per chiarire le parti di codice più ambigue o spiegarne la funzione.

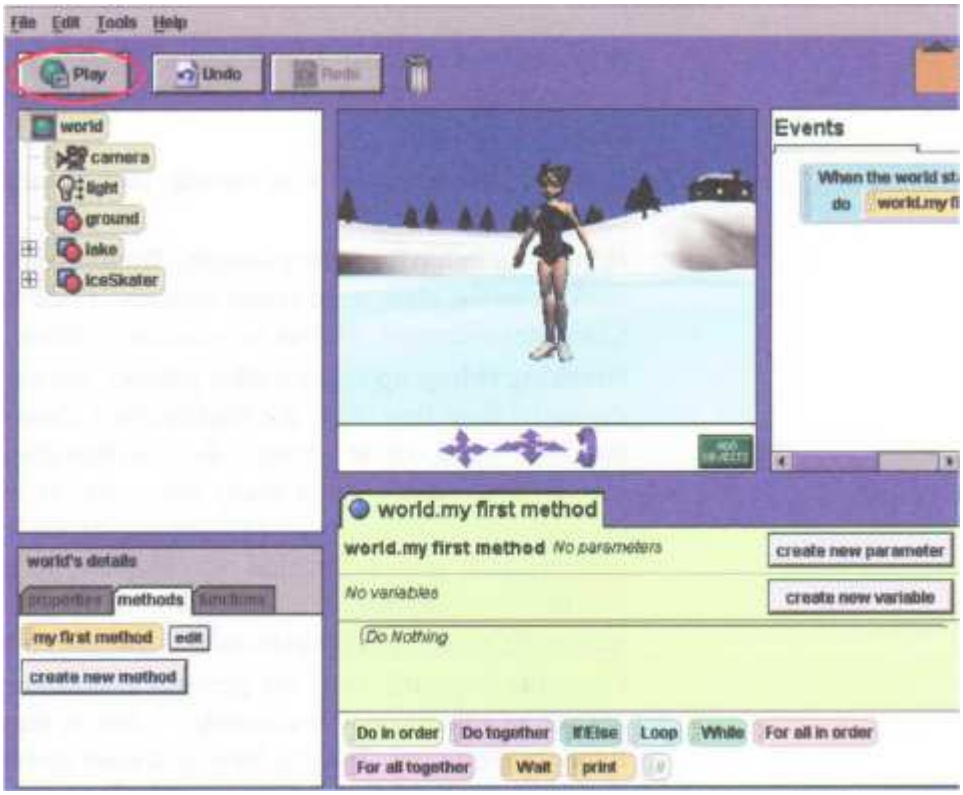


Figura 4.1: Interfaccia di Alice 2

4.2 Descrizione dell'interfaccia

Per quanto riguarda l'interfaccia, Alice 2 è molto simile ad Alice 3 per cui nella trattazione successiva verranno richiamati solo i concetti principali ed evidenziate solo le differenze più significative e meno intuitive rispetto alla nuova versione.

Imparare a programmare in Alice vuol dire saper realizzare dei mondi virtuali 3D inserendovi oggetti e personaggi, lasciando spazio alla creatività, e successivamente realizzare dei programmi per animare tali mondi in modo che raccontino una storia come un film o siano interattivi come dei videogiochi.

Un mondo virtuale inizia con un modello per una scena iniziale. I modelli sono mostrati nella finestra di apertura all'avvio di Alice (figura 4.2) e rappresentano lo sfondo o ambiente relativo al mondo virtuale.

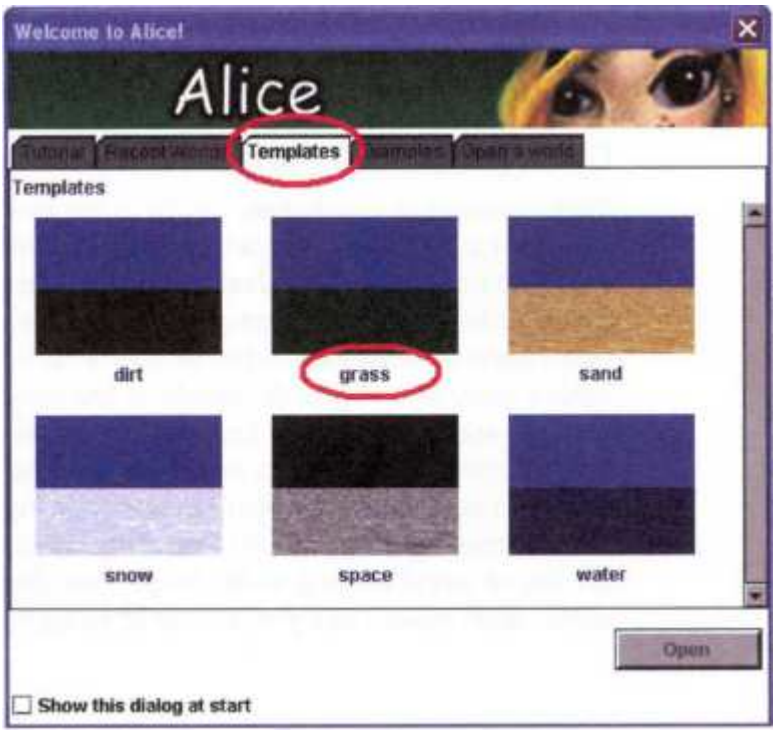


Figura 4.2: Modelli per il mondo virtuale

Per quanto riguarda l'inserimento di altri oggetti nel mondo "spoglio" così creato, in Alice vi è una vasta galleria di modelli 3D che si possono scegliere (figura 4.3). È sufficiente premere il pulsante verde "Add Objects", selezionare l'oggetto scelto dalla galleria e trascinarlo adeguatamente nel mondo virtuale (e alla fine premere il pulsante verde "Done" per tornare alla schermata principale). Bisogna comunque tenere presente che Alice non è un programma di grafica 3D e quindi non ha in sé gli strumenti per creare dei modelli per nuovi oggetti (presenta solo gli editor "hebuilder" e "shebuilder" per creare le persone a piacimento), quindi è molto probabile che non si riesca a realizzare completamente il mondo desiderato poiché non è possibile rappresentare ogni cosa immaginabile da chiunque.

Gli oggetti in Alice sono a tre dimensioni ed ogni oggetto ha larghezza, altezza, profondità, orientazione e può essere mosso (in sei possibili direzioni o gradi di libertà), ridimensionato e ruotato a piacimento. Inoltre come in Alice 3 gli oggetti possono essere rinominati, purché non vengano violati i vincoli già visti nell'ultima versione (a parte il vincolo relativo agli spazi nel nome, che qui non è presente). Anche qui è possibile inserire immagini 2D o testo 3D.

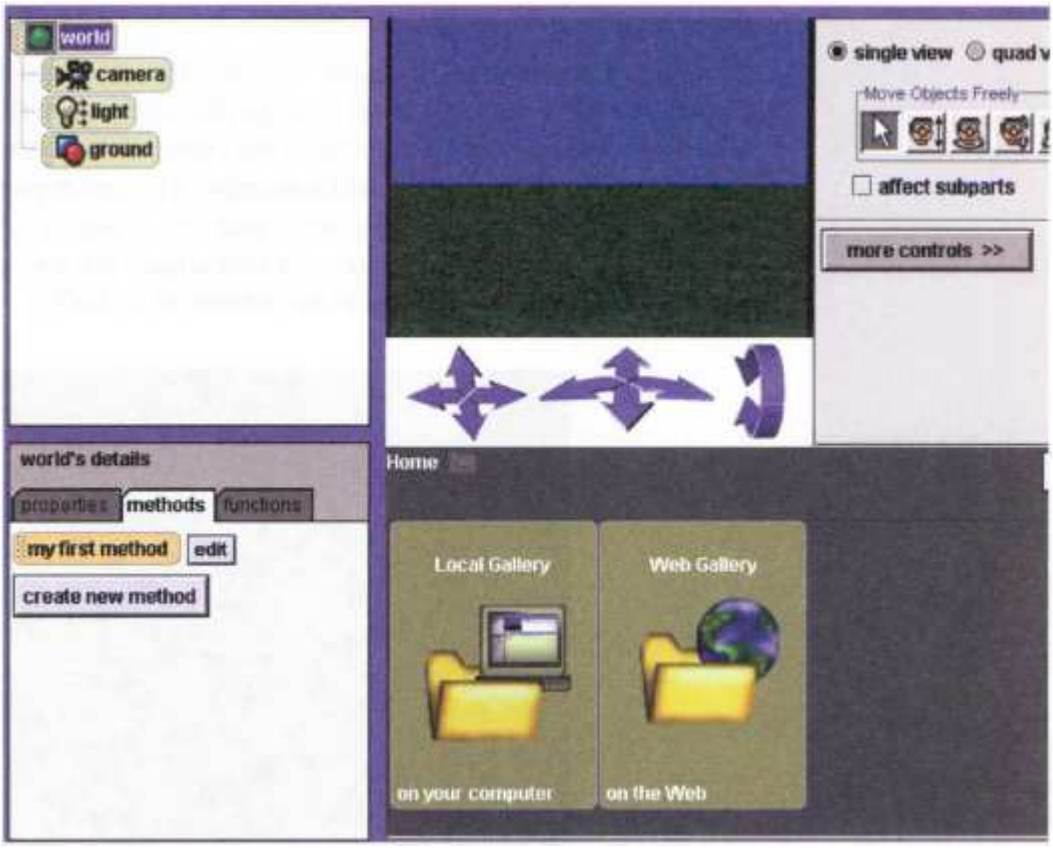


Figura 4.3: Galleria per gli oggetti 3D

Un altro concetto importante in Alice è la posizione degli oggetti, poiché essa è relativa al centro del mondo che è l'origine dei tre assi coordinati (il terreno è automaticamente posizionato al centro del mondo come mostrato in figura 4.4).

Una differenza rispetto ad Alice 3 di cui ci si accorge subito è la presenza di diversi tutorial che vengono proposti non appena si avvia Alice 2 per la prima volta: essi spiegano in breve le caratteristiche principali, come inserire oggetti e codice, creare i

propri metodi e testare il funzionamento dei programmi creati. Ciò è molto importante perché risulta molto più utile ai fini dell'apprendimento applicarsi un po' nella pratica piuttosto che concentrarsi unicamente su guide teoriche dispersive (che comunque poi sarebbe meglio leggere). In questo modo si facilita e velocizza l'assimilazione dei concetti e, data la semplicità dei tutorial, si evita di spaventare gli studenti alle prime armi, al contrario incoraggiandoli.

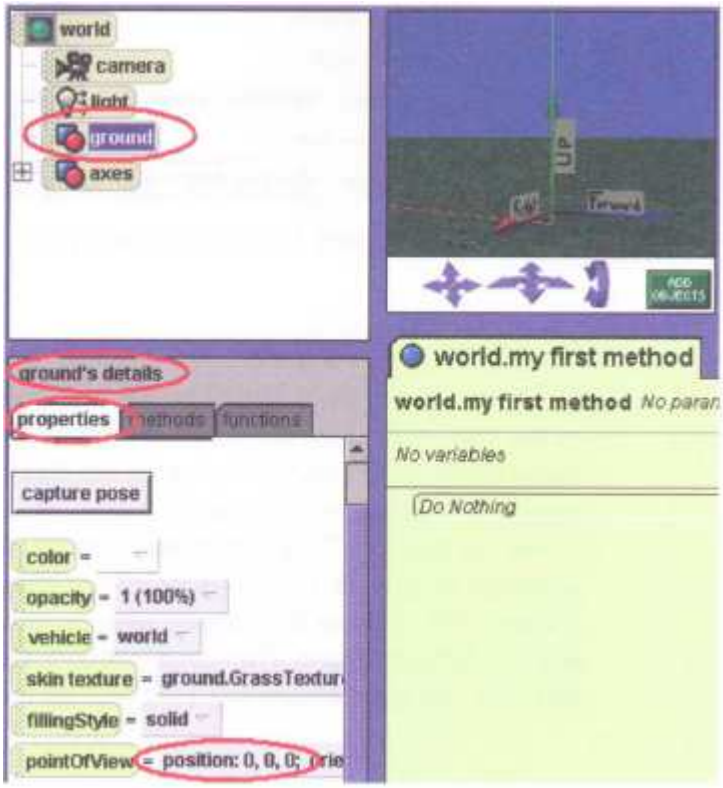


Figura 4.4: Posizione degli oggetti relativa al mondo

Un'altra differenza rispetto ad Alice 3 è che in Alice 2 l'opzione "vehicle" per ogni oggetto è impostata sull'oggetto stesso, quindi quando si muove il terreno tutti gli oggetti, anche quelli appoggiati sopra, rimangono immobili e cambiano solamente le coordinate del terreno rispetto al centro del mondo mentre in Alice 3 tale opzione è impostata sulla scena stessa, per cui muovendola vengono mossi tutti gli oggetti.

4.3 Introduzione alla programmazione

Una volta inseriti tutti gli oggetti e costruita la propria scena iniziale (procedendo in maniera analoga a quanto visto per Alice 3), bisognerà iniziare a scrivere il codice per animarla, cioè “dare vita” ai personaggi. Per farlo basterà utilizzare il “Code Program Editor” nella finestra principale di Alice 2, evidenziato in figura 4.5.

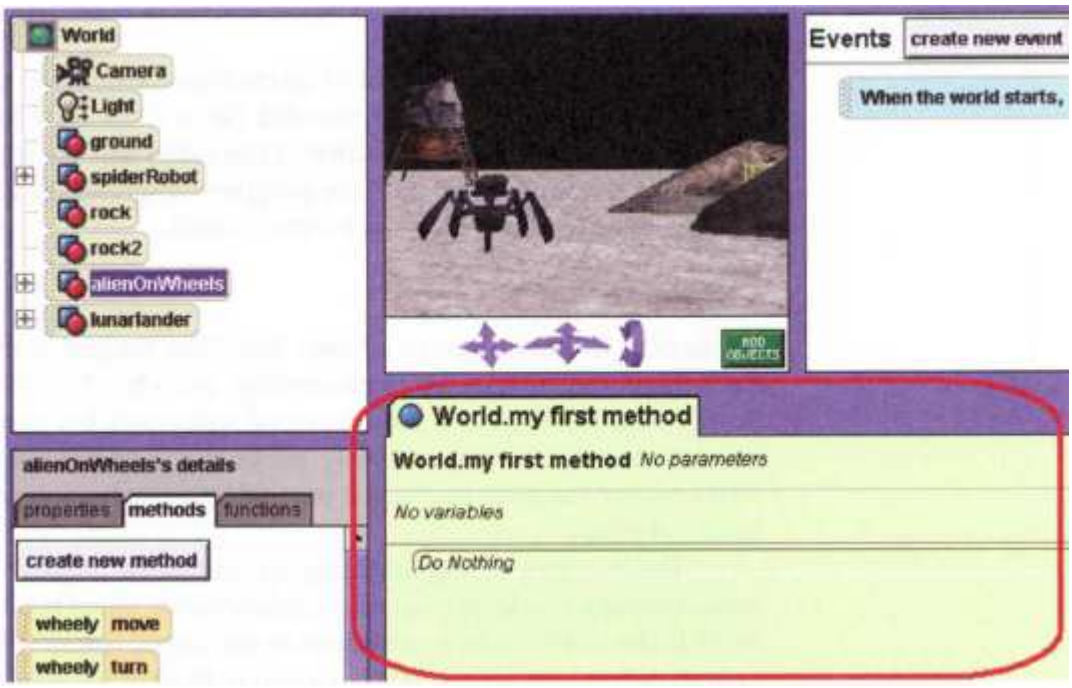


Figura 4.5: Code Program Editor

Di default il programma che uno studente realizza in Alice 2 si chiama “my first method”, ma naturalmente può essere rinominato utilizzando il pannello in basso a sinistra nella finestra principale, che specifica i dettagli e le proprietà degli oggetti. È bene prima di iniziare a scrivere il programma, fare una traccia di ciò che si vuole rappresentare nella scena ed un elenco delle azioni da compiere. Fatto ciò si può iniziare a scrivere il programma utilizzando il comodo ambiente “drag and drop” di Alice. Come detto in precedenza il codice può essere visualizzato secondo la sintassi di Alice oppure secondo una sintassi per alcuni aspetti simile a quella di Java

(ricordiamo che in quest'ultimo caso Alice 2 ha una sintassi molto meno simile a Java rispetto ad Alice 3). Per visualizzare i metodi disponibili per un oggetto bisogna selezionare l'oggetto dal pannello in alto a sinistra nella finestra principale ed essi verranno mostrati nel pannello in basso a sinistra: in figura 4.5 è stato selezionato l'oggetto "aliensOnWheels" e nella parte sottostante si possono vedere i suoi metodi, proprietà o funzioni. Una volta stabilito quale azione far compiere ad un oggetto è sufficiente trascinare il metodo desiderato nel Code Program Editor . Come in Java ed altri linguaggi di programmazione i metodi possono avere dei parametri: vi sono dei parametri che bisogna necessariamente inserire (ed in quel caso Alice obbliga l'utente a sceglierli come mostrato in figura 4.6) oppure parametri che si possono specificare successivamente cliccando sul pulsante "more" nella riga relativa a quel metodo nell'editor. È ovviamente possibile modificare tutti i parametri quante volte si vuole tramite l'editor (anche inserendo valori personalizzati diversi dalle liste di valori che compaiono), compreso il parametro relativo all'oggetto su cui viene applicato quel metodo (sempre che vi siano altri oggetti nella scena che hanno tra gli altri anche quel metodo stesso).

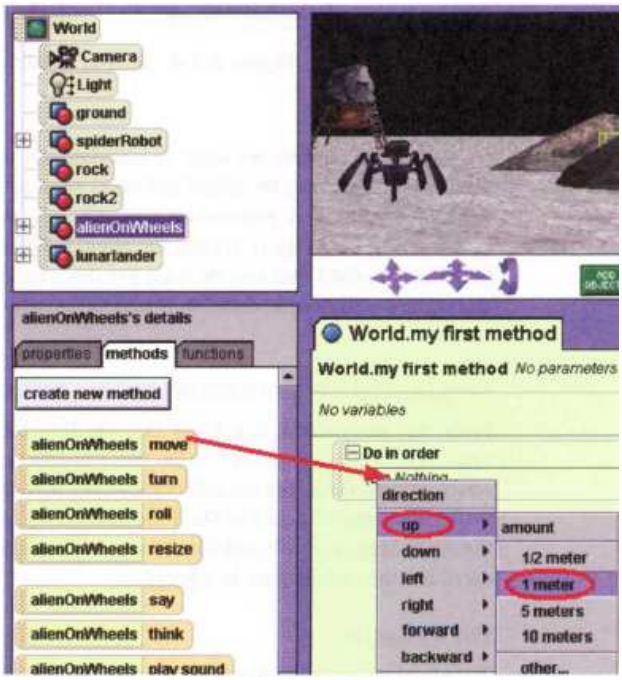


Figura 4.6: Inserimento di un metodo

Come in Alice 3, gli oggetti sono costituiti da sottoparti, che presentano anch'esse dei metodi che possono eseguire. Inoltre anche qui è possibile modificare le proprietà degli oggetti durante l'esecuzione: è sufficiente trascinare la proprietà richiesta all'interno del Code Program Editor nel punto desiderato perché venga modificata in quel preciso momento. Anche le funzioni relative agli oggetti possono essere utilizzate all'interno del codice poiché infatti i loro valori di ritorno possono risultare molto utili nella realizzazione del programma.

Un'altra cosa molto importante nella scrittura di un programma è l'ordine con cui devono essere eseguite le azioni: esse possono essere eseguite in sequenza seguendo un ordine, simultaneamente, ripetute in un ciclo oppure eseguite solo se si verificano determinate condizioni (istruzioni condizionali). La scelta di default al momento della creazione del programma, al di fuori delle strutture o se non vi sono altre strutture inserite è "Do in order", cioè le operazioni vengono eseguite in sequenza secondo l'ordine dall'alto verso il basso nel Code Program Editor. È possibile inserire le altre modalità di esecuzione, chiamate strutture di controllo, trascinandole dalla parte inferiore del Code Program Editor come mostrato in figura 4.7 ed è inoltre possibile innestarle una dentro l'altra effettuando molteplici combinazioni ed ottenendo effetti diversi.

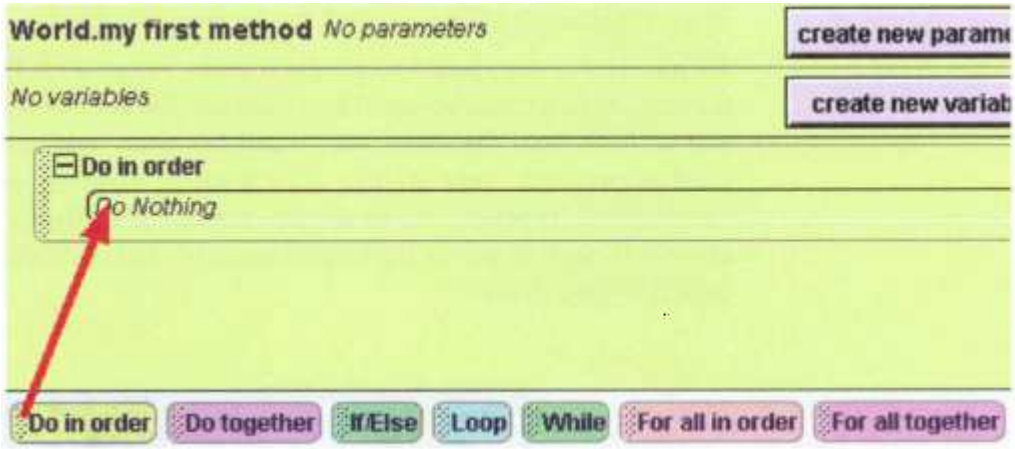


Figura 4.7: Ordine delle operazioni

Tra le altre cose è possibile inserire anche dei commenti all'interno del programma, cosa molto utile per far capire ad altre persone a cosa serve una determinata parte di codice, come mostrato in figura 4.8.

Figura 4.8: Inserimento di commenti

Come già detto, la posizione e l'orientazione degli oggetti sono concetti di cui è necessario tener conto, poiché non conoscendone i meccanismi è facile incorrere in errori. Se si fa muovere un oggetto in una direzione tramite un metodo ci si accorge dell'importanza di tali concetti: infatti esso si muove (per esempio a sinistra) non secondo l'osservatore della scena bensì rispetto al centro e all'orientazione dei suoi assi coordinati. Non tutti gli oggetti hanno gli assi coordinati disposti nello stesso modo, per cui può essere utile usare il metodo "orient to" (sia prima di scrivere il codice che come istruzione da eseguire) che serve a fare in modo che un oggetto abbia gli assi disposti nello stesso modo di un altro oggetto. Molto utile è anche il metodo "set vehicle to" che fa in modo che un oggetto sia "veicolato" ad un altro, cioè si muova esattamente come quest'ultimo. Tra i parametri dei metodi di movimento vi è inoltre "asSeenBy" che fa in modo di utilizzare l'orientamento di un oggetto per guidare il movimento di un altro oggetto, cioè la direzione del movimento non è più quella propria dell'oggetto stesso ma quella dell'altro oggetto scelto. Importanti nella costruzione della scena sono anche i metodi "turn to face", grazie al quale un oggetto si gira in modo da averne di fronte un altro selezionato, e "point at" con il quale si può allineare il centro di un oggetto con il centro di un altro. Per quanto riguarda quest'ultimo metodo bisogna tener presente che se i due centri si trovano ad altezze diverse la direzione che li unisce sarà obliqua e di conseguenza

applicando questo metodo su un oggetto esso verrà inclinato. Per evitare ciò, è possibile impostare a “true” il valore di un ulteriore parametro di questo metodo, cioè “onlyAffectYaw” , rendendo così il risultato identico a quello del metodo “turn to face”.

Riassumendo alcuni concetti detti in precedenza, in Alice, come nei classici linguaggi di programmazione, vi è la possibilità di utilizzare:

- Istruzioni, che servono per far compiere agli oggetti delle azioni.
- Strutture di controllo quali:
  - “do in order” che serve per eseguire le istruzioni in sequenza.
  - “do together” che serve per eseguire le istruzioni simultaneamente .
  - “if/else” che serve per eseguire istruzioni condizionali.
  - “loop” che serve per ripetere le istruzioni un certo numero di volte, anche casuale o infinito, ma comunque noto.
  - “while” che serve per ripetere le istruzioni finché una certa condizione è verificata.
  - “for all in order” che, una volta creata una lista di elementi (come ad esempio oggetti, numeri, valori booleani, stringhe ed altri oggetti di vario tipo) oppure selezionatane una preesistente, permette di svolgere operazioni con essi, uno alla volta.
  - “for all together” che, una volta creata una lista di elementi (come ad esempio oggetti, numeri, valori booleani, stringhe ed altri oggetti di vario tipo) oppure selezionatane una preesistente, permette di svolgere operazioni con essi simultaneamente.
- Funzioni, che servono per ottenere il valore di determinate variabili o parametri (di solito relativi ad un oggetto) oppure sono delle espressioni logiche che ritornano un valore booleano.
- Espressioni, cioè operazioni matematiche su un numero o su altri tipi di variabili.

Non tutte le proprietà degli oggetti sono riportate nell'apposito pannello (in basso a sinistra) nella schermata principale, ma solo quelle che comunemente vengono modificate con maggior frequenza nella costruzione di una scena. Per quanto riguarda le altre proprietà è sufficiente "chiedere ad Alice": nello stesso pannello, cliccando sul pulsante "functions", comparirà una lista di domande o funzioni per ottenere determinati valori relativi alle caratteristiche dell'oggetto selezionato, informazioni relative al suo stato o sulle relazioni con altri oggetti. Le funzioni sono raggruppate in diverse categorie:

- "Proximity": sono quelle funzioni che dicono quanto è distante un oggetto rispetto ad un altro.
- "Size": queste funzioni restituiscono valori come altezza, larghezza e profondità di un oggetto, oppure comparano le dimensioni dell'oggetto con quelle di altri oggetti.
- "Spatial Relation": servono per comparare l'orientazione di un oggetto con quella di altri oggetti.
- "Point of view": ridanno la posizione rispetto al mondo.
- "Other": altre funzioni varie, come il nome di una sottoparte di un oggetto.

Le funzioni di solito restituiscono quattro tipi di valori a seconda della "domanda che pongono": un numero, un valore booleano (vero o falso), una stringa oppure un oggetto. Per questo motivo possono essere inserite come parametri di alcuni metodi in modo tale da utilizzare a proprio vantaggio i loro valori di ritorno: per esempio si potrebbe voler far muovere un oggetto in una certa direzione e fargli percorrere la stessa distanza che c'è tra esso ed un altro oggetto come mostrato in figura 4.9.

In questo caso è da notare che la distanza è misurata dal centro di un oggetto al centro di un altro, quindi se non si usa la funzione in maniera adeguata è facile che si verifichino delle collisioni (non sempre volute), cioè che un oggetto "entri nell'altro", poiché occupa parte del suo stesso spazio fisico. Per evitare le collisioni è anche

possibile aggiustare la distanza utilizzando le espressioni matematiche fornite da Alice. Sono disponibili addizione, sottrazione, moltiplicazione e divisione.
Per utilizzarle è sufficiente selezionarle dal relativo menù accanto al valore appena selezionato, come mostrato in figura 4.10.

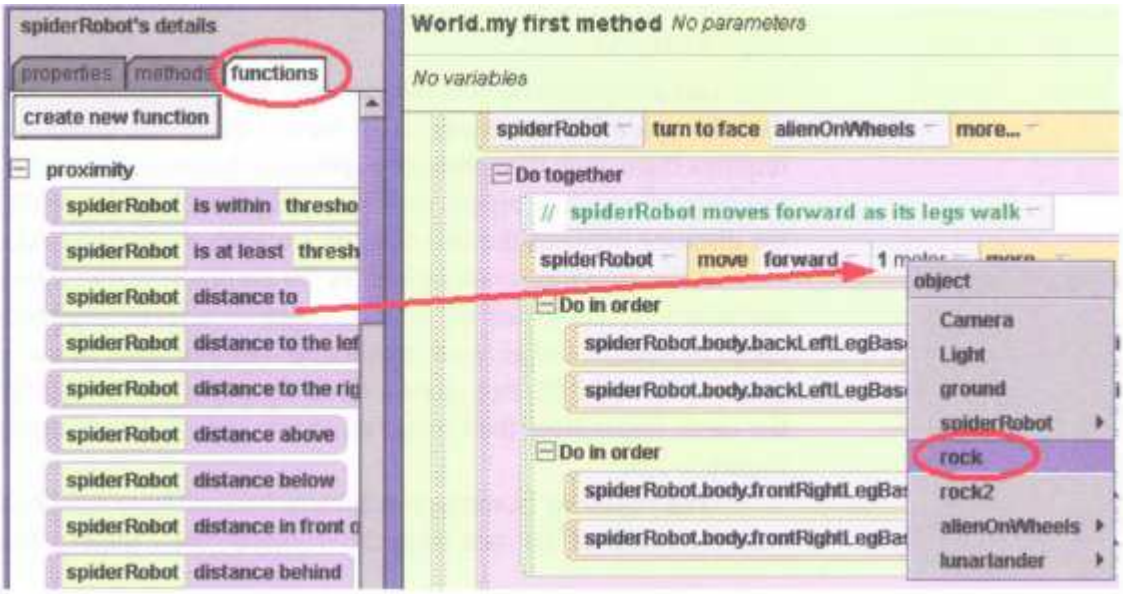


Figura 4.9: Uso di una funzione

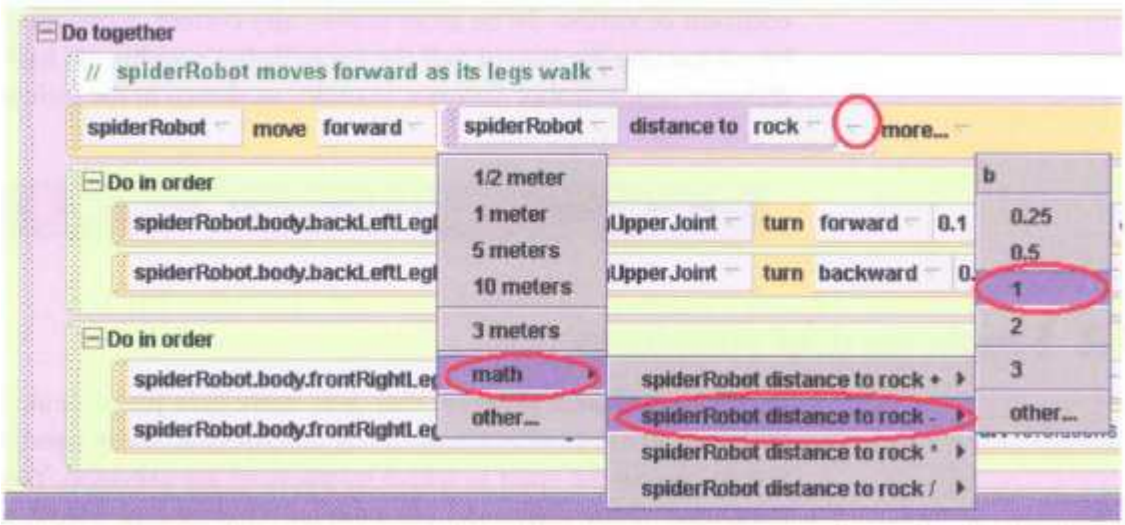


Figura 4.10: Uso di un operatore matematico

In figura 4.11 invece è mostrato come utilizzare un operatore matematico con un valore ritornato da una funzione.

Molto frequente è anche l'uso di funzioni che ritornano un valore booleano, poiché esse vengono usate nelle strutture di controllo "if/else" per eseguire espressioni condizionali oppure nei cicli"while" per eseguire la ripetizione di una parte di codice finché una certa condizione è verificata.

Figura 4.11: Uso di un operatore matematico con una funzione

Selezionando come oggetto l'intero mondo (dall'albero degli oggetti in alto a sinistra), è possibile notare che tra le funzioni che compaiono vi sono anche gli operatori relazionali (ma vi sono anche operatori logici o di altro tipo) che permettono di verificare ulteriori condizioni (figura 4.12). Per utilizzarli è necessario prima inserire dei valori numerici (o booleani per quanto riguarda gli operatori logici) e poi sostituirli con funzioni più adatte, come già fatto in precedenza, poiché ricordiamo che Alice impone di inserire immediatamente i parametri fondamentali per un metodo, impedendo così che vengano commessi degli errori di sintassi.

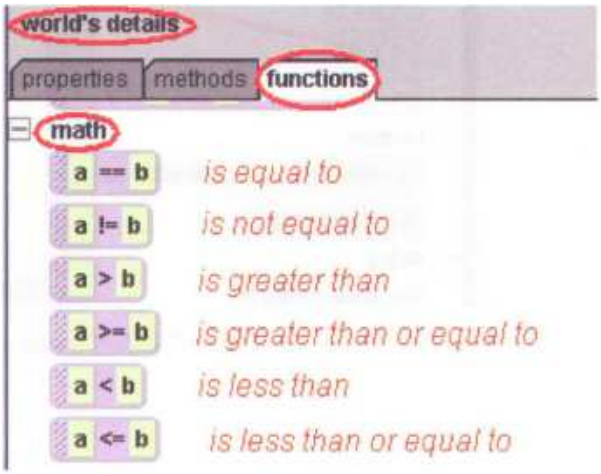


Figura 4.12: Operatori relazionali

Per quanto riguarda il controllo della telecamera, l'analisi è molto simile a quella fatta per Alice 3. Leggermente diversa è invece la gestione dei punti di vista multipli della telecamera (anche se il concetto rimane lo stesso), creati tramite i "dummy objects", cioè degli oggetti fittizi che memorizzano una certa posizione, e quindi un punto di vista, per la telecamera. Per spostare la telecamera (prima o durante l'esecuzione) è sufficiente utilizzare il metodo "set point of view to" che porta la telecamera nella posizione del dummy object selezionato (vi è anche il pulsante "move camera to dummy", presente in figura 4.13) oppure di qualsiasi altro oggetto. Per creare un punto di vista per la telecamera è sufficiente entrare nella modalità per aggiungere oggetti, cliccare sul pulsante "more controls" (vedi figura 4.13) nella parte destra e successivamente selezionare il pulsante "drop dummy at camera" (che al momento della creazione salva la posizione attuale della telecamera come sua posizione) oppure il pulsante "drop dummy at selected object" (che al momento della creazione salva la posizione attuale dell'oggetto selezionato come sua posizione). Ogni volta che si clicca su uno di quei pulsanti viene creato un dummy object: tutti questi oggetti fittizi sono poi raggruppati in un'unica cartella "Dummy Objects" nella parte sinistra della finestra (nell'albero degli oggetti). È inoltre possibile fare in modo che la telecamera segua il movimento di qualsiasi oggetto (prima o durante l'esecuzione) semplicemente cambiando la proprietà "vehicle" come per un qualsiasi altro oggetto. È utile notare che inserire nel codice delle istruzioni per muovere la telecamera in maniera adeguata durante l'esecuzione, può creare degli effetti particolarmente graditi o focalizzare l'attenzione su alcuni dettagli o personaggi, per cui è da tenere fortemente in considerazione come possibile tecnica di animazione.

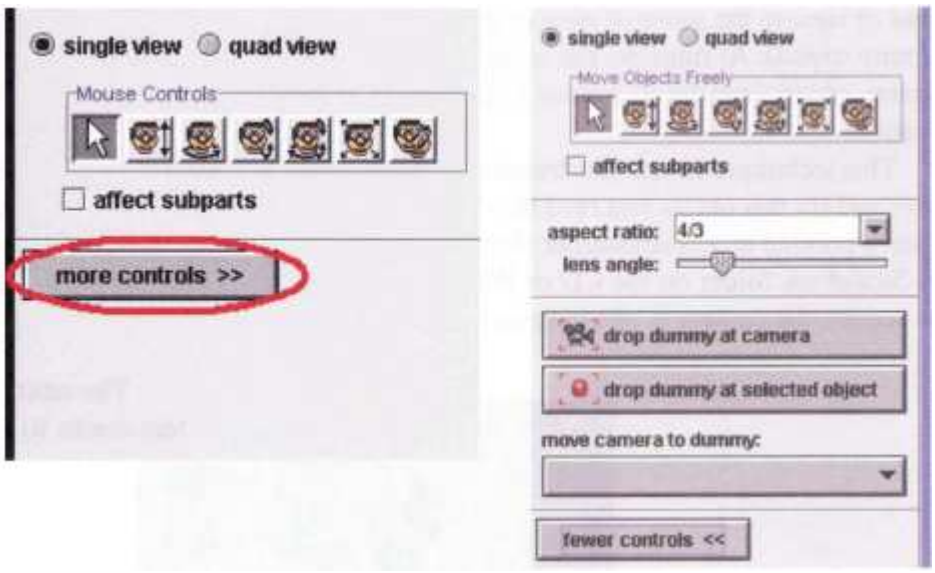


Figura 4.13: Punti di vista multipli per la telecamera

4.4 Classi, oggetti, funzioni, metodi e parametri

Come nei linguaggi di programmazione orientati agli oggetti, anche in Alice 2 sono presenti diverse classi (non mostrate nell'editor a differenza di Alice 3) che definiscono particolari tipi di oggetti (che quindi sono istanze delle classi) che utilizziamo per animare la scena. Come già detto nell'analisi relativa ad Alice 3, si applicano le stesse consuetudini presenti in Java come per esempio distinguere classi ed oggetti rappresentando le prime con lettera iniziale del nome maiuscola e gli ultimi con lettera iniziale nel nome minuscola (poi non è proibito rinominare un oggetto con lettera iniziale maiuscola, basta farlo consapevolmente). Stessa cosa per quanto riguarda le regole sintattiche come per esempio non avere nomi di oggetti uguali (le regole sono le stesse già viste per Alice 3, a parte la mancanza del vincolo sugli spazi all'interno di un nome). Le classi, in Alice come anche in Java, raggruppano oggetti che condividono alcune caratteristiche e metodi.
È inoltre possibile creare più metodi diversi e complessi (come nei linguaggi di programmazione orientati agli oggetti) che contengono parti di codice del

programma, sia perché suddividere un vasto programma in più parti serve per semplificarne la costruzione, sia per renderlo più leggibile e magari evitare di ripetere alcune lunghe parti di codice. Per fare ciò si utilizzano i metodi primitivi o di base forniti da Alice, che messi insieme ne possono creare di più complessi.
Per creare un nuovo metodo è necessario selezionare l'intero mondo dall'albero degli oggetti in alto a sinistra nella finestra principale (quindi il metodo creato sarà un metodo "globale", cioè relativo al mondo, ovvero tutto il programma), cliccare sul pulsante "create new method" nella parte in basso a sinistra e dare un nome al metodo (che per prassi dovrebbe iniziare con la lettera minuscola) come mostrato in figura 4.14. Una volta creato il nuovo metodo Alice lo apre automaticamente nell'editor accanto agli altri metodi aperti in precedenza.

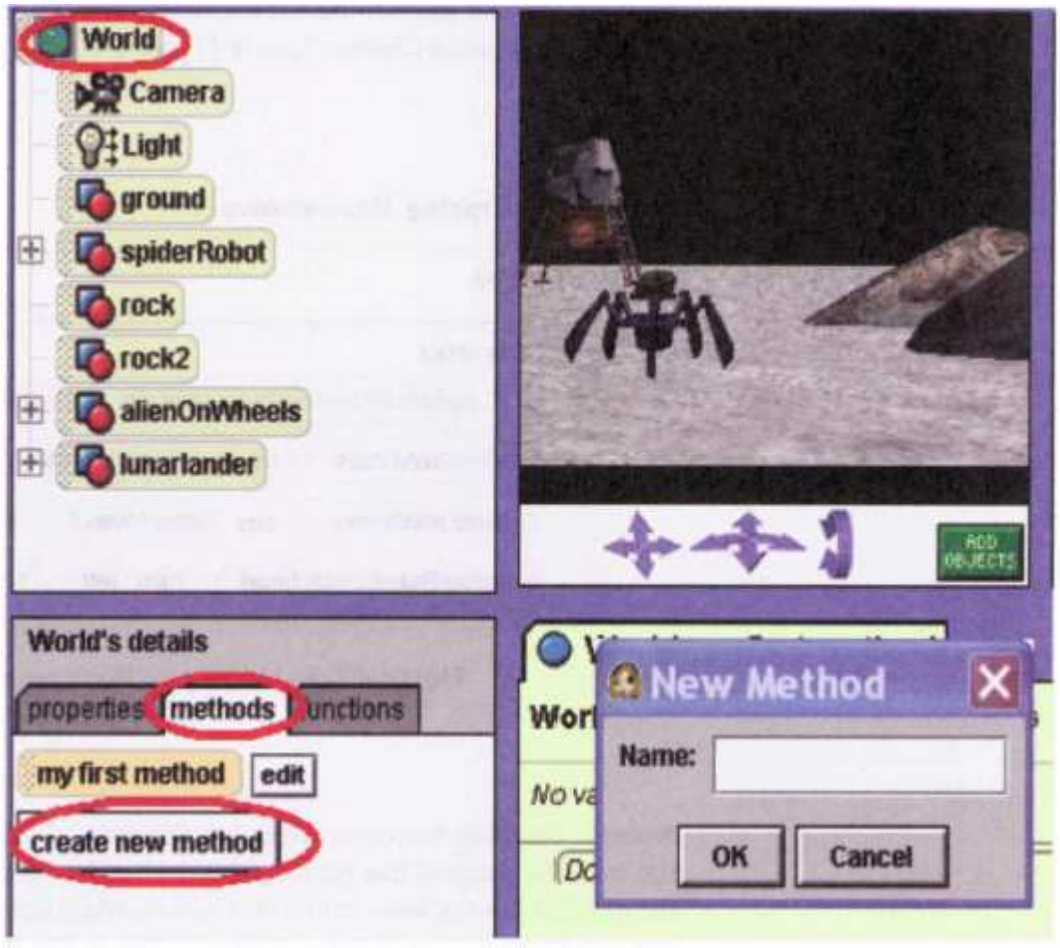


Figura 4.14: Creazione di un nuovo metodo

Per eseguire il metodo appena creato vi sono varie alternative: si può utilizzare "Events editor" in alto a destra nell'interfaccia di Alice, inserendo un'istruzione che comporti la sua esecuzione (per esempio impostando che all'avvio del programma venga avviato quel metodo al posto del metodo di default "my first method"), oppure è possibile trascinare il metodo (dallo stesso pannello con cui è stato creato, cioè quello in basso a sinistra in figura 4.14) in "my first method" o in qualunque altro metodo che venga già eseguito all'avvio del programma. Se invece si vuole eliminare un metodo (che non sia predefinito) lo si può fare trascinandolo dal pannello in basso a sinistra al cestino che si trova in alto a sinistra, facendo attenzione però ad aver eliminato prima tutte le chiamate a quel metodo all'interno del programma, poiché in caso contrario Alice ne impedirebbe l'eliminazione dato che sarebbe fonte di errore. Come già detto, un altro aspetto importante riguardante i metodi è quello relativo ai parametri da essi richiesti: ve ne sono di necessari (che Alice obbliga ad inserire) ed altri il cui uso è facoltativo, ma essi sono tutti molto utili poiché danno ai metodi informazioni molto importanti per funzionare al meglio e svolgere i compiti richiesti. È anche possibile, oltre ad inserire dei parametri specifici, creare dei nuovi parametri, che possono servire per molteplici scopi, cliccando sul pulsante "create new parameter" nel Code Program Editor. In particolare mantenendo un parametro oggetto generico è possibile evitare di dover riscrivere più volte lo stesso codice: se ci sono più oggetti in metodi diversi che devono compiere le stesse azioni è sufficiente creare un parametro oggetto generico (che non è un vero e proprio oggetto) all'interno di un metodo (che conterrà quindi il parametro generico), trascinare quante volte si vuole il metodo con il parametro generico all'interno del metodo che verrà eseguito, scegliendo ogni volta l'oggetto che deve essere inserito al posto del parametro generico. Bisogna però prestare attenzione poiché, non essendo un vero e proprio oggetto, se ci si dimentica di fare questa sostituzione ed il metodo che contiene il parametro generico è invocato in esecuzione, premendo il tasto "Play" per far partire il programma verrà segnalato un errore in esecuzione ed Alice ricorderà che la sostituzione non è stata effettuata. Un esempio di tutto ciò è riportato in figura

4.15: nell'esempio "commonMethod" è il metodo il cui codice andrebbe ripetuto mentre "genericObjectParameter" è il parametro oggetto generico creato in questo caso, che appartiene al metodo (e ciò lo si vede poiché è scritto accanto al nome del metodo) e che viene in esso utilizzato. Si vede quindi che in "my first method", che è il metodo che poi verrà eseguito, il metodo "commonMethod" è riportato quattro volte, ma al parametro generico sono stati sostituiti quattro veri oggetti appartenenti al mondo che devono compiere quella stessa azione. Ovviamente si può fare ciò anche con altri tipi di parametri, non solo oggetti, ed inoltre Alice permette di inserire molteplici tipi di parametri che possono essere utilizzati in svariati modi.

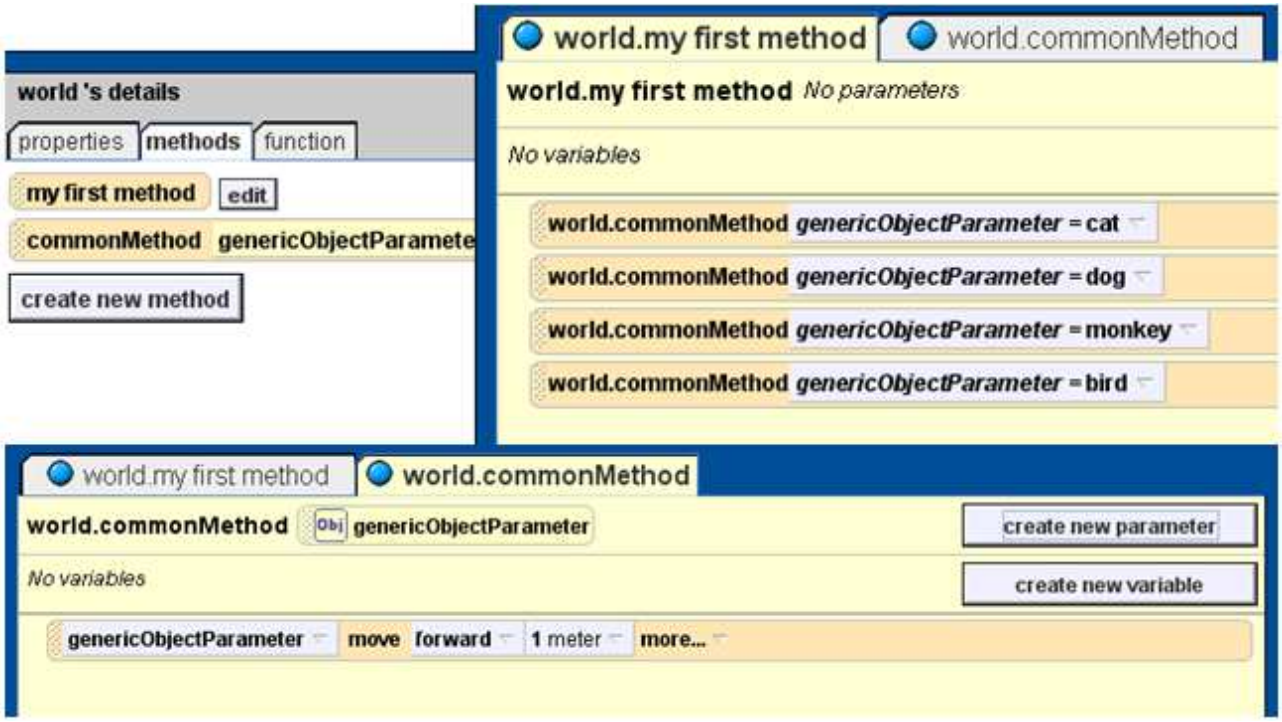


Figura 4.15: Uso di un parametro oggetto generico

Un altro concetto relativo ai linguaggi di programmazione è quello dell'ereditarietà, anche se in Alice 2 non è propriamente implementato. Infatti, sebbene a partire da una delle classi degli oggetti 3D definiti nella galleria, sia possibile creare una sottoclasse che "condivida" gli stessi metodi della superclasse e contenga anche dei nuovi metodi da noi definiti utilizzando quelli basilari, quelli creati risultano essere degli oggetti distinti e non "derivati" da una certa classe (pur condividendone la

maggior parte dei metodi). In ogni caso, per fare ciò, dopo aver inserito l'oggetto nel mondo, bisogna selezionarlo nell'albero degli oggetti e cliccare sul pulsante "create new method": è da notare che il metodo che viene creato è un metodo di classe, non più relativo al mondo ed infatti si capisce ciò anche osservando che è chiamato con il nome dell'oggetto ed il nome del metodo separati da un punto (per esempio "iceSkater.skate"). Una volta completati, i nuovi metodi e l'oggetto devono essere salvati, per cui per prima cosa bisogna cambiare nome all'oggetto dall'albero degli oggetti (perché in questo caso non si vuole sovrascrivere la classe modificata ma salvarla come una nuova). Ora è sufficiente cliccare col tasto destro del mouse sull'oggetto appena rinominato dall'albero degli oggetti e scegliere "save object" dal menù. Una volta salvato l'oggetto dove si desidera, lo si potrà utilizzare in qualsiasi programma senza dover riscrivere il codice dei metodi appena creati e potendo comunque invocare anche i metodi che erano utilizzabili dall'oggetto di classe superiore (poiché il nuovo oggetto risulta di fatto una copia modificata). Per poterlo utilizzare sarà sufficiente aprire il menù "File" in alto a sinistra nella finestra principale di Alice, selezionare "Import" e quindi l'oggetto appena creato dalla posizione in cui era stato salvato. Nella creazione di questi metodi di classe è consigliabile non includere degli altri oggetti appartenenti al mondo, poiché non è detto che saranno sempre presenti, e questo potrebbe creare degli errori in esecuzione in altre versioni di Alice 2. In questo caso conviene utilizzare la tecnica già vista relativa al parametro oggetto generico, che quindi permette di scrivere del codice che funzionerà in qualsiasi versione, semplicemente sostituendo il parametro con un oggetto esistente in quella versione. Stesso discorso per quanto riguarda i suoni: è meglio importarli dall'esterno piuttosto che fare affidamento su quelli interni ad Alice, poiché in questo modo verranno salvati insieme alla nuova classe e non si rischia di incorrere in errori in caso siano assenti in altre versioni del programma. Altra cosa che bisogna evitare è quella di inserire nei metodi di classe dei metodi relativi al mondo, poiché in altre versioni potrebbero non essere presenti.

Un altro strumento molto importante in Alice sono le funzioni, poiché permettono di verificare il valore di alcuni parametri o avere informazioni sugli oggetti durante esecuzione del programma. Inoltre esse a differenza dei metodi che sono progettati per creare delle animazioni, non modificano lo stato del mondo, ma devono semplicemente ritornare un valore. Oltre all'insieme di funzioni predefinite di Alice, è possibile crearne di proprie: è sufficiente selezionare l'oggetto per il quale si vuole creare la funzione dall'albero degli oggetti, selezionare "functions" nel pannello in basso a sinistra nella finestra principale, e premere il pulsante "create new function". In questo modo è anche possibile salvare poi la nuova classe creata avente una funzione in più come fatto per quanto riguarda i metodi. Se invece la funzione riguarda più oggetti è conveniente creare la funzione selezionando il mondo dall'albero degli oggetti. Come mostrato in figura 4.16, nell'editor relativo alla funzione appena creata, è già presente l'istruzione "return" che serve per ritornare un valore e che Alice impedisce di rimuovere poiché sarebbe causa di errore.
Nella parte inferiore dell'editor, le strutture di controllo "do in order", "do together", "for all together" e "wait" non sono più presenti, ma al loro posto è possibile trascinare nell'editor altri "return".

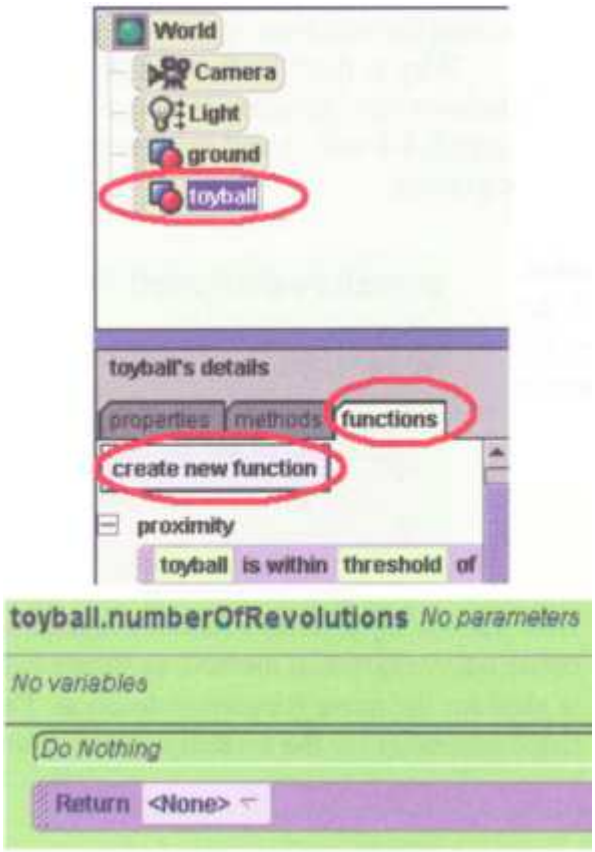


Figura 4.16: Creazione di una nuova funzione

In Alice è abilitato anche un altro importante strumento: la ricorsione (cioè un metodo che chiama se stesso all'interno di esso). Questo è uno strumento molto potente grazie al quale si possono scrivere programmi complessi. Purtroppo però non è facile da utilizzare e va usato con cautela perché è spesso fonte di errori. Infatti non appena si trascina, dall'apposito pannello, un metodo dentro se stesso, Alice avvisa prontamente l'utente di quello che sta facendo e gli chiede di confermare se lo sta facendo consapevolmente. Per alcuni aspetti l'uso della ricorsione da parte di persone che non hanno mai avuto l'occasione di studiare un linguaggio di programmazione potrebbe creare confusione, e gli stessi programmatori di Alice, che comunque l'hanno abilitato, si rendono conto che possa costituire un problema per un utente inesperto.

4.5 Gli eventi e la loro gestione

In Alice, come già visto, è possibile creare programmi per animare una scena come una sorta di video o film che si può stare a guardare. Però Alice da anche la possibilità di creare programmi interattivi o semplici videogiochi in grado di rispondere all'input che l'utente trasmette con il mouse o da tastiera e la cui sequenza di azioni non è sempre la stessa ma è decisa durante l'esecuzione. Ogni input fornito dall'utente è per il programma un evento, in base al quale deve o meno eseguire una determinata parte di codice, se ciò è stato precedentemente definito nell'insieme delle istruzioni in esso contenute. Un altro tipo di evento è quello riguardante le azioni degli oggetti: per esempio in un programma vi possono essere degli elementi di casualità, come l'uso di un valore "random" (casuale), a seconda dei quali possono essere eseguite alcune istruzioni o altre oppure eseguite in maniera ogni volta diversa. Per poter creare e quindi gestire degli eventi per un programma interattivo è necessario premere il pulsante "create new event" nell'editor degli eventi e scegliere innanzitutto il tipo di evento (come mostrato in figura 4.17), e poi il metodo che deve essere eseguito.

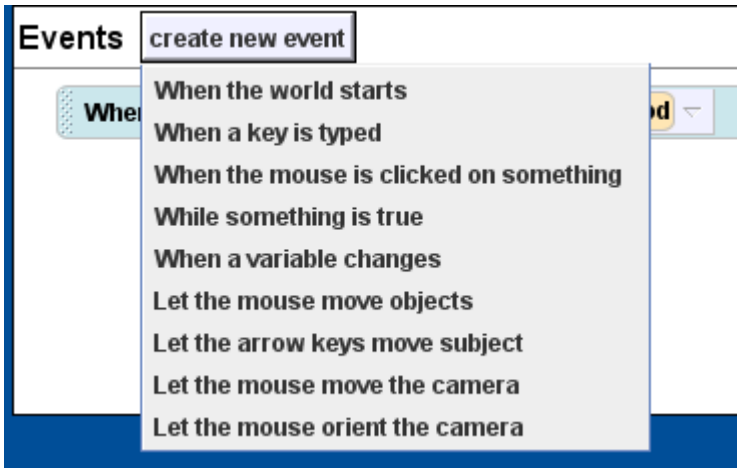


Figura 4.17: Creazione di un nuovo evento

L’istruzione di default inserita da Alice è tale da dire al programma di eseguire “my first method” all’avvio stesso del programma. Ma come si nota dalla figura 4.17 vi sono molte altre opzioni tra gli eventi, a cui si possono applicare tutti i metodi desiderati:

- “When the world starts” permette di scegliere un metodo da eseguire all’avvio del programma (anche questo evento può anche essere inserito più volte come per quanto riguarda i successivi).
- “When a key is typed” permette di scegliere il metodo da far eseguire ogni volta che, durante l’esecuzione, viene premuto il pulsante desiderato (o qualsiasi pulsante) sulla tastiera.
- “When the mouse is clicked on something” permette di scegliere il metodo da far eseguire ogni volta che, durante l’esecuzione, si clicca con il mouse su uno degli oggetti del mondo (o una sua sottoparte).
- “While something is true” crea un ciclo “while” in cui è possibile inserire una condizione e scegliere tre metodi da far eseguire all’inizio del ciclo (“Begin”), durante il ciclo (“During”) ed alla fine del ciclo (“End”). Il metodo inserito all’inizio del ciclo viene eseguito una sola volta quando si entra nel ciclo, il metodo inserito alla voce “During” viene eseguito (quindi ripetuto) finché la condizione scelta all’inizio non diventa falsa (anche se si trova nel bel mezzo dell’esecuzione del ciclo, non appena la condizione diventa falsa, il programma vi esce) mentre il metodo scelto per la fine del ciclo viene eseguito una sola volta non appena si esce dal ciclo. Se durante l’esecuzione tale condizione è falsa (o lo diventa), e vi è la possibilità che essa ritorni ad essere vera (magari tramite l’input di un utente) il programma rimane in attesa che essa cambi il suo stato e non appena diventa vera tutto il ciclo “while” viene di nuovo eseguito.
- “When a variable changes” permette di scegliere il metodo da far eseguire ogni volta che, durante l’esecuzione, una variabile scelta cambia valore.

Tra gli eventi vi sono anche delle opzioni che non sono dei veri e propri eventi, ma abilitano il movimento della telecamera o di altri oggetti tramite il mouse o le frecce come si vede anche dalla figura 4.17.

4.6 Creazione di un programma

Come fatto per Alice 3, si vuole anche qui mostrare dal punto di vista pratico come costruire un programma utilizzando Alice 2. Per quanto riguarda la storia, scegliamo di costruirne una diversa rispetto a quella del programma creato in Alice 3, poiché il codice sarebbe molto simile e ripetitivo. Una volta costruita la scena di partenza con le tecniche elencate nelle sezioni precedenti, si può iniziare anche in questo caso a scrivere il codice. Anche qui vogliamo rappresentare il codice prima con la sintassi di Alice 2 e successivamente con quella simile a Java, sebbene essa non sia accurata come quella simile a Java di Alice 3. Avendo già ampiamente presentato gli elementi basilari che servono alla costruzione di un programma in Alice 2, il codice seguente non verrà analizzato dettagliatamente come fatto per Alice 3, ma verranno evidenziati solamente gli aspetti di maggiore interesse.

Questa volta la storia è un po' diversa, e ne verranno analizzate solo le parti di codice più significative. Si tratta del classico tema della principessa in pericolo che deve essere salvata: in questo caso una principessa è minacciata da un feroce drago. La principessa chiede aiuto poiché il drago le sta girando attorno; successivamente compare un cavaliere che sconfigge il drago con un potente calcio. Una volta salvata la principessa, i due si baciano e partono verso il castello del cavaliere. La scena di partenza è rappresentata in figura 4.18. Da notare la maggior qualità grafica dello sfondo rispetto ai personaggi: questo perche lo sfondo non è altro che un'immagine 2D importata con le modalità viste nelle sezioni precedenti.

In figura 4.19 è riportato il codice relativo al metodo “my first method”. Si nota già dal nome del metodo che in Alice 2 è possibile assegnare nomi contenenti degli spazi, cosa non possibile né in Alice 3 né in Java. Anche qui sono stati creati dei metodi per dividere il programma: alcuni di essi sono specifici per degli oggetti, altri invece sono stati creati come metodi globali (appartenenti a tutto il mondo), poiché contengono dei riferimenti a degli oggetti del mondo stesso. I metodi relativi al mondo sono “calcio_al_drago” e “bacio”, mentre per la principessa è stato creato solo il metodo “chiede_aiuto” e per il drago “muove_le_ali”. Per quanto riguarda il cavaliere vi era già un metodo predefinito, e cioè “AnimateBreathing”, che viene utilizzato nell’ultima riga del metodo “calcio_al_drago” (figura 4.21).

Figura 4.18: Scena iniziale del programma

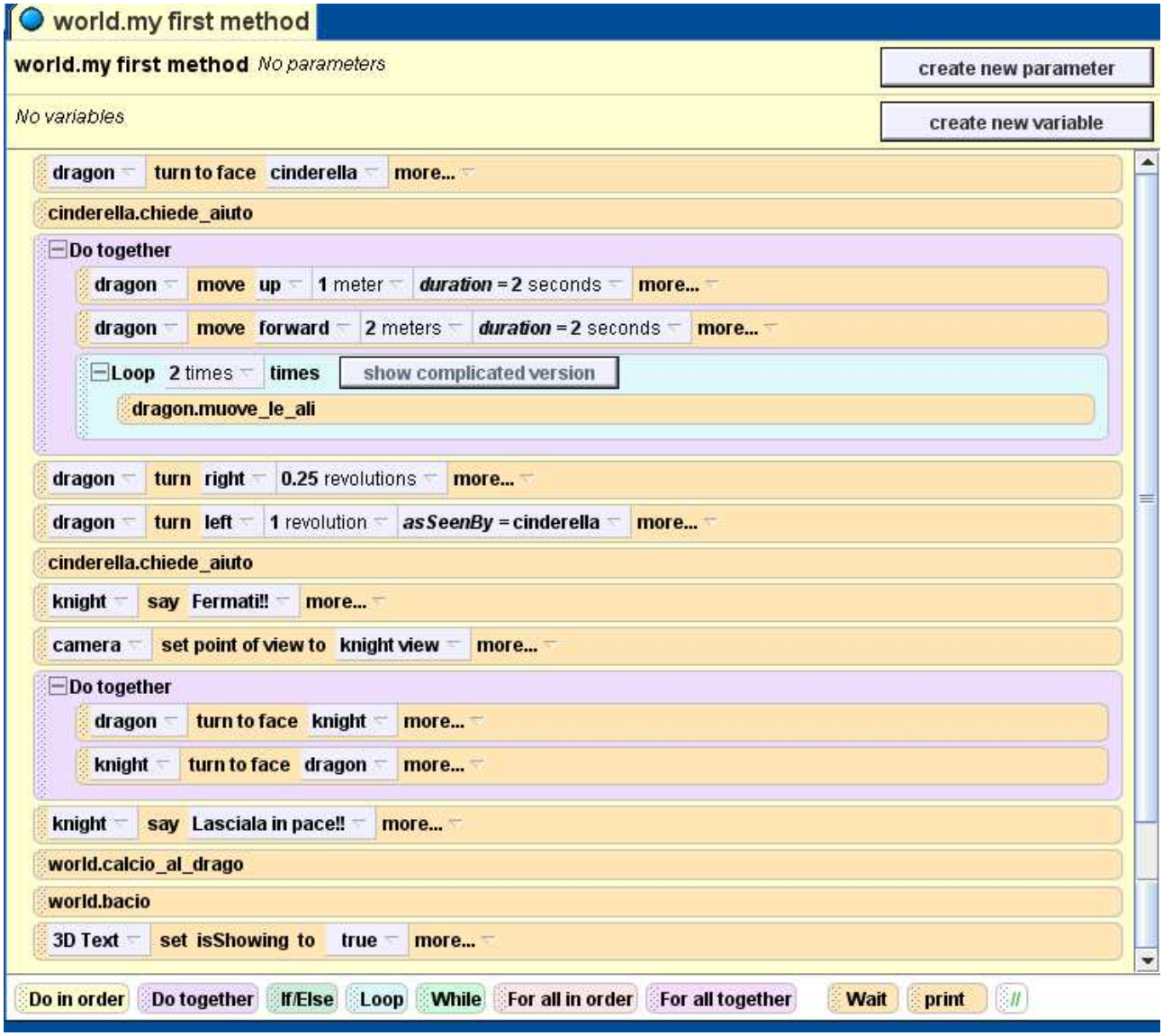


Figura 4.19: Codice relativo al metodo “my first method”

Figura 4.20: Codice presente nell’editor degli eventi.

In figura 4.20 vi è il codice relativo agli eventi: come già spiegato, la prima istruzione serve per far eseguire “my first method” all’avvio del programma.

La seconda serve per fare in modo che il drago esegua il metodo “muove_le_ali” ogni volta (ed in qualsiasi momento durante l’esecuzione del programma) che l’utente preme il tasto “space” sulla tastiera. Il fatto che l’utente possa premere spazio anche quando il drago è stato sconfitto, non è necessariamente da gestire in questo caso, poiché il drago per prima cosa viene allontanato dalla scena, ed inoltre viene portata la sua opacità al valore 0.0 (penultima riga del codice in figura 4.21) in modo da nasconderlo: quindi, qualora l’utente decidesse di fargli muovere ancora le ali, non si avvertirebbe nessun cambiamento nella scena essendo il drago invisibile.
Da notare che in Alice 2 vi sono due modi per rendere un oggetto invisibile: impostare a 0.0 la sua opacità, come fatto in questo caso, oppure portare al valore “false” la proprietà “isShowing”, proprietà utilizzata anche nell’ultima riga di codice di figura 4.19 per rendere visibile (portando il valore da “false” a “true”) il testo 3D che compare per comunicare all’utente la fine del racconto.
Le altre strutture di controllo ed i metodi con le loro proprietà, presenti nel codice di figura 4.19, sono già stati ampiamente descritti nelle sezioni precedenti, per cui non verranno analizzati. Stessa cosa per quanto riguarda il codice degli altri metodi, riportati nelle figure di seguito.

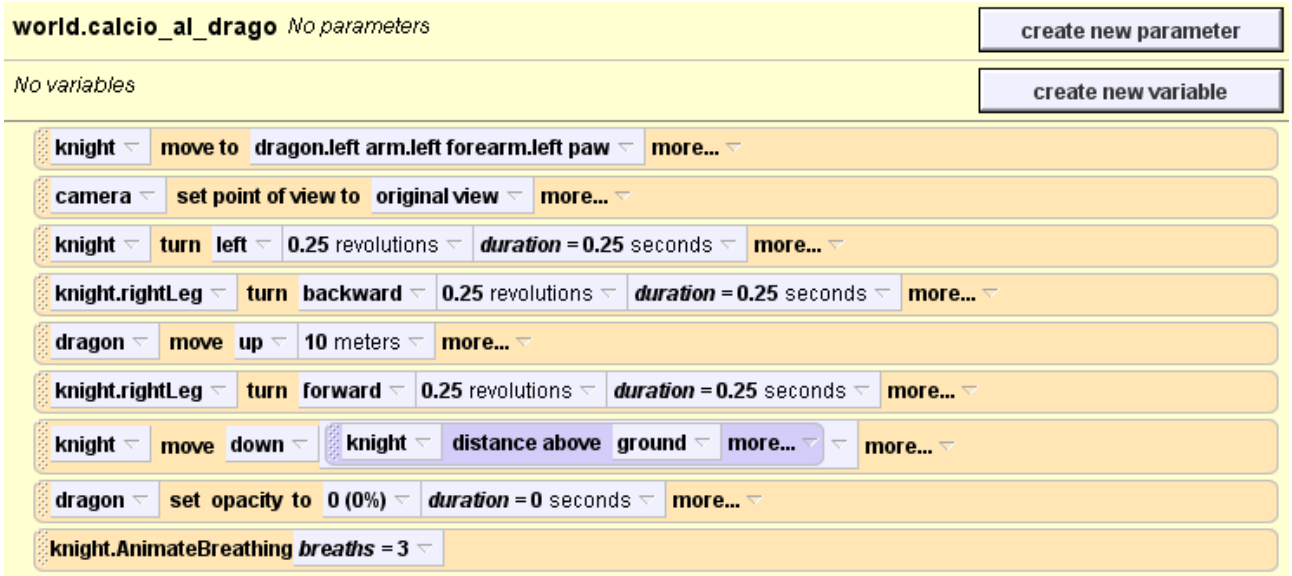


Figura 4.21: Codice relativo al metodo “calcio_al_drago”

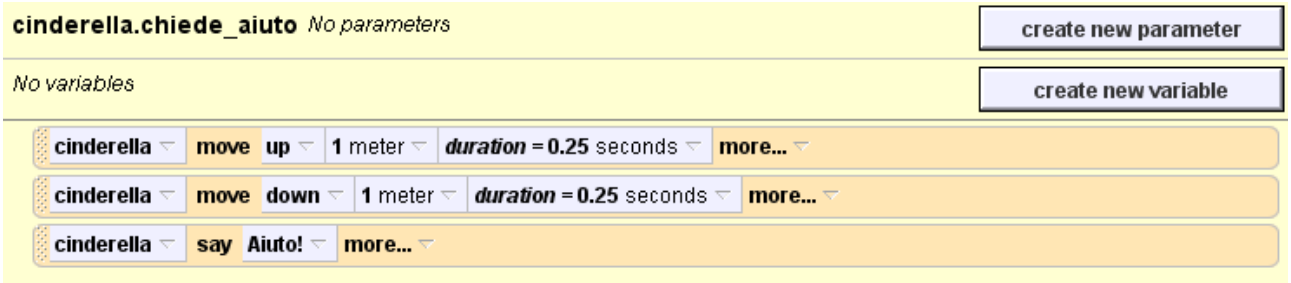


Figura 4.22: Codice relativo al metodo “chiede_aiuto”

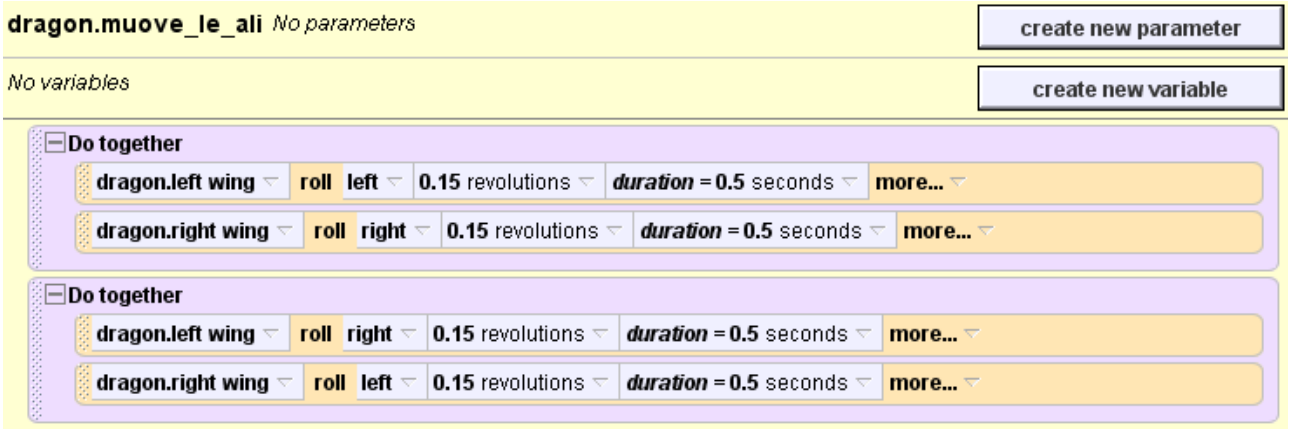


Figura 4.23: Codice relativo al metodo “muove le ali”

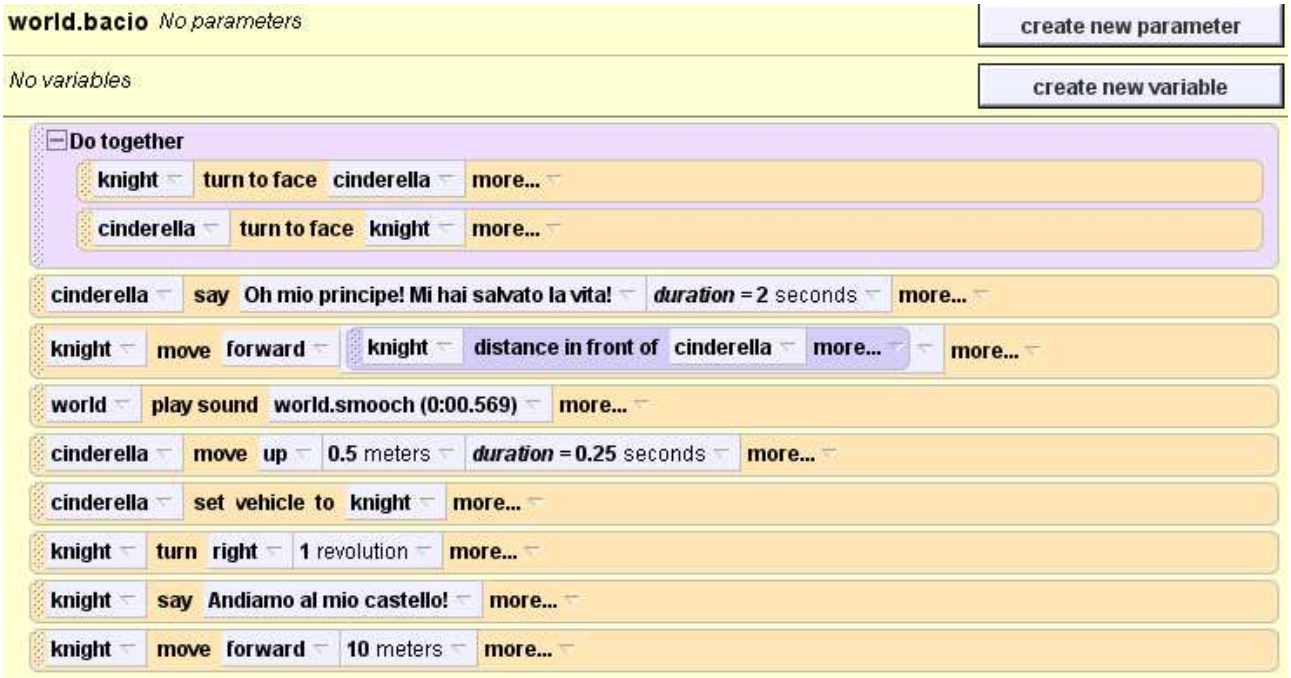


Figura 4.24: Codice relativo al metodo “bacio”

Da notare l'uso di funzioni che ritornano un valore (in questo caso numerico, cioè una distanza), come parametro passato ai metodi nella settima riga del codice di figura 4.21 e nella quarta riga di quello presente in figura 4.24. Sempre nella quinta riga di quest'ultima figura è da notare l'introduzione di un suono (relativo al bacio), che è fatto riprodurre dal mondo.

Infine, per completezza ed in linea con quanto fatto per Alice 3, sono riportate di seguito le parti dello stesso programma appena creato, visualizzate però con la sintassi simile a Java di Alice 2.

Figura 4.25: Codice "Java-like" presente nell'editor degli eventi.

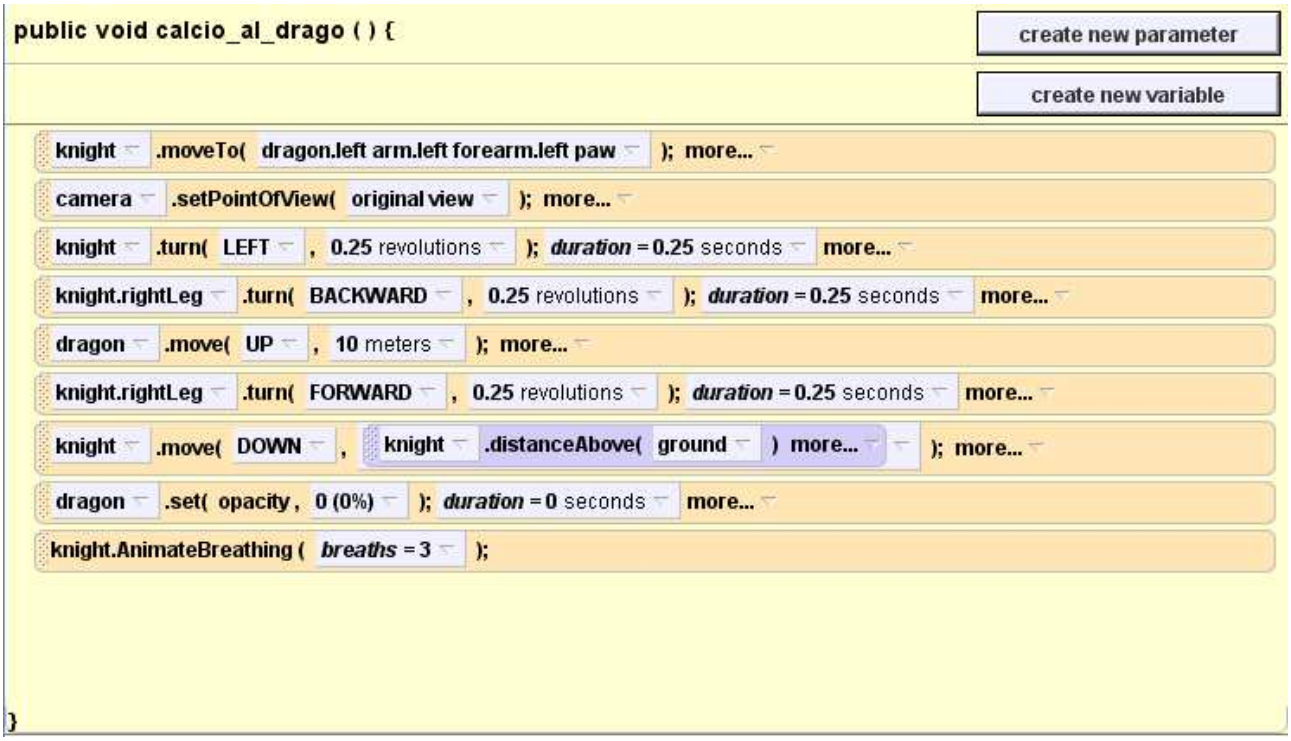


Figura 4.26: Codice "Java-like" relativo al metodo "calcio_al_drago"

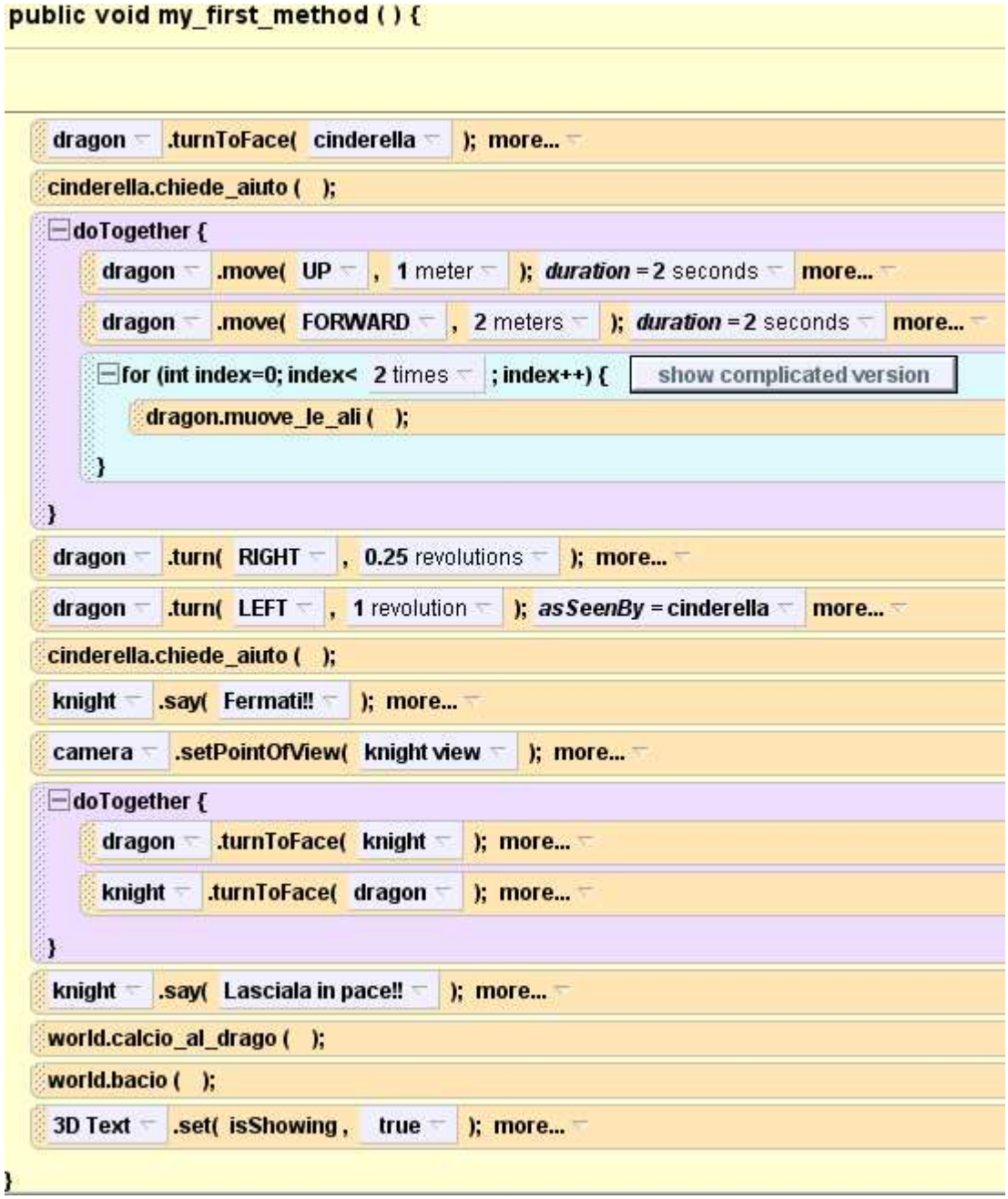


Figura 4.27: Codice “Java-like” relativo al metodo “my first method”

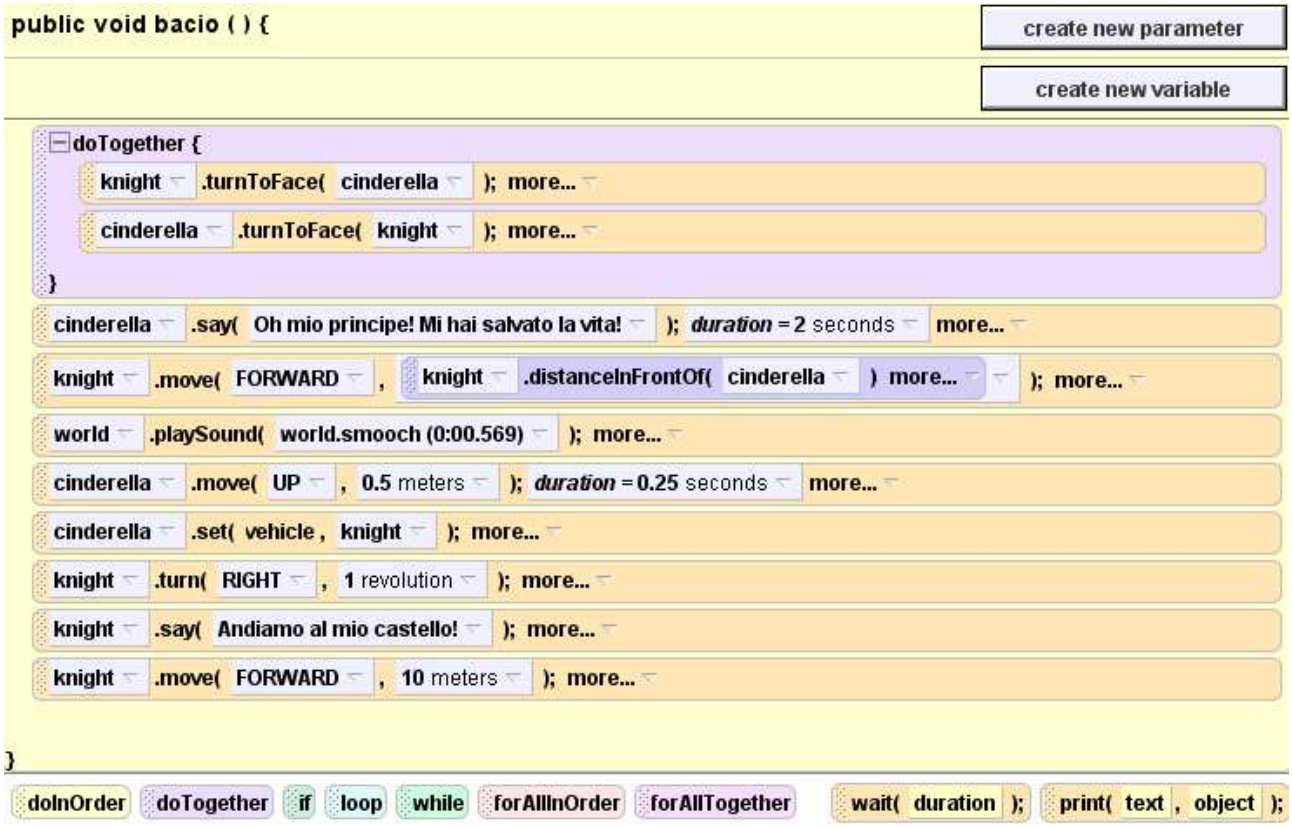


Figura 4.28: Codice “Java-like” relativo al metodo “bacio”

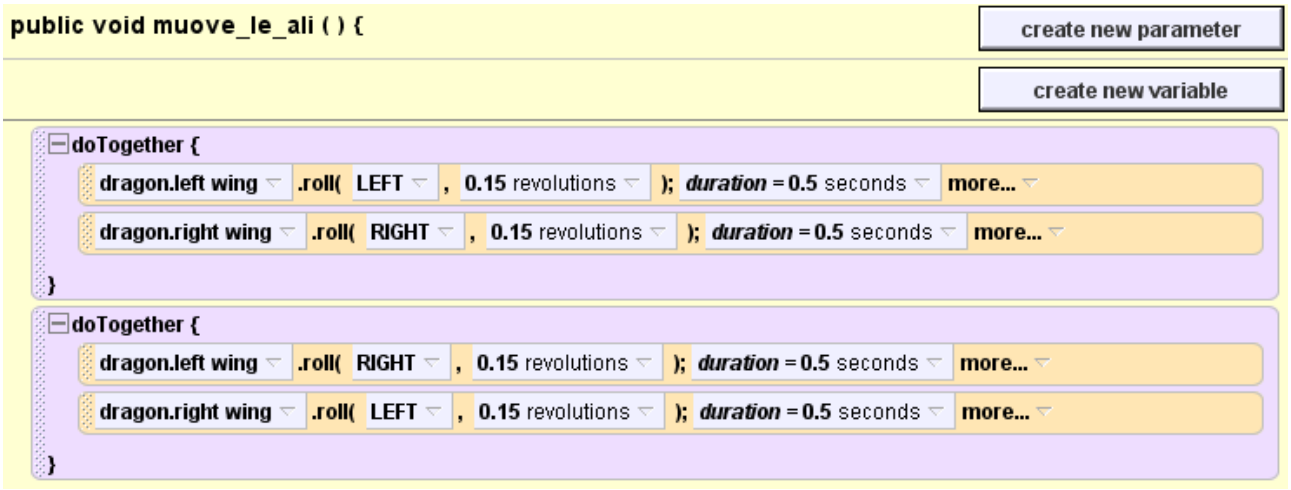


Figura 4.29: Codice “Java-like” relativo al metodo “muove_le_ali”

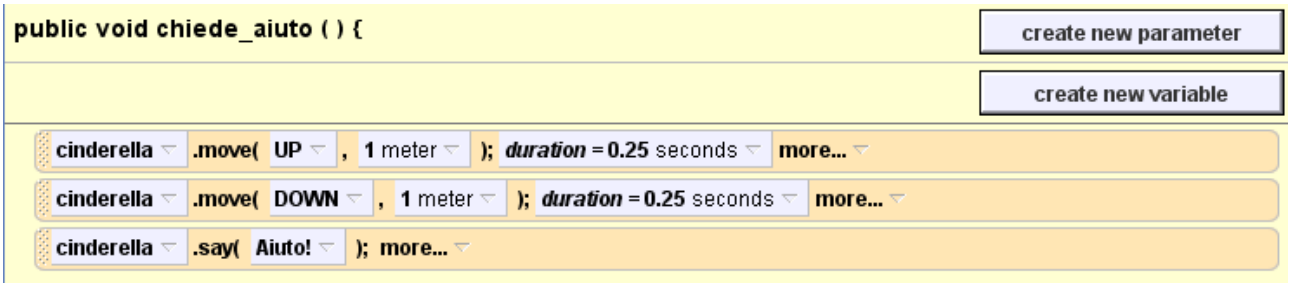


Figura 4.30: Codice “Java-like” relativo al metodo “chiede_aiuto”

5 STORYTELLING ALICE

5.1 Introduzione

Le donne sono attualmente sottorappresentate nel campo dell'informatica: per questo si vuole cercare di invertire questa tendenza aumentandone il numero, poiché ciò porterebbe numerosi vantaggi quali il miglioramento della tecnologia e la diversificazione dei punti di vista che influenzano il design e la tecnologia stessa. Per questo motivo, a partire da Alice 2, è stato creato Storytelling Alice, un software ideato appositamente per far avere alle ragazze una prima esperienza positiva con la programmazione. Lo sviluppo di Storytelling Alice è stato possibile grazie a numerosi test effettuati con la collaborazione di più di 250 ragazze della scuola media. Per determinare le cose di cui avevano bisogno le ragazze per creare le loro storie, esse sono state osservate mentre utilizzavano Alice 2 e lo stesso Storytelling Alice nelle sue prime versioni, analizzando con attenzione i programmi sviluppati. Per incoraggiare le ragazze a creare i particolari tipi di storie che erano in grado di immaginare, Storytelling Alice include animazioni di alto livello che rappresentano numerose interazioni sociali trai personaggi, una galleria di oggetti 3D disegnati secondo i gusti delle ragazze in modo da stimolare ed incentivare la produzione di storie, numerosi tutorial progettati per far prendere confidenza con l'ambiente di programmazione alle ragazze. Per determinare il successo nell'insegnamento della programmazione focalizzando l'interesse delle ragazze nella produzione di storie, sono stati condotti degli studi comparando l'esperienza vissuta dalle ragazze che hanno utilizzato Alice 2 con quella delle loro coetanee che invece hanno usato Storytelling Alice. Anche per via della somiglianza trai due software, dal punto di vista dell'apprendimento dei concetti basilari della programmazione, sono stati ottenuti gli stessi risultati per entrambe le versioni. La cosa più rilevante però è stata che le ragazze che hanno utilizzato Storytelling Alice, hanno speso molto più tempo

nella programmazione rispetto alle loro coetanee, chiedendo addirittura del tempo ulteriore per poter completare con successo i loro progetti. Quindi sebbene Storytelling Alice non sia più supportato, i risultati che sono stati ottenuti durante la sperimentazione non sono assolutamente da trascurare, e devono fungere da esempio per progetti didattici futuri.

5.2 Differenze ed innovazioni rispetto ad Alice 2.x

Storytelling Alice, soprattutto per quanto riguarda l'interfaccia, si presenta come un software estremamente simile ad Alice 2. Infatti esso è stato sviluppato a partire da una versione di Alice 2, a cui sono state aggiunte nuove funzioni e modificati alcuni aspetti. Per questo sarebbe ripetitivo descrivere nuovamente l'interfaccia e gli altri elementi di Alice 2, ma è sicuramente utile elencare le modifiche più significative effettuate al software che lo rendono più interessante.

Innanzitutto è stato realizzato un insieme di animazioni particolari, specifiche per i tipi di azioni che comunemente compiono i personaggi dei racconti. Infatti con i metodi presenti in Alice 2 non è immediato riuscire a costruire degli abbinamenti tali da simulare queste animazioni. Per esempio se si vuole far camminare un personaggio o un animale, ma non è presente un metodo apposito, bisogna necessariamente utilizzare delle combinazioni di metodi elementari, e fare ampio uso delle sottoparti degli oggetti, per ottenere un effetto simile. Chi ha sviluppato Storytelling Alice era ben conscio delle difficoltà che possono avere persone senza esperienza di programmazione, nell'abbinare metodi generici per ottenerne di più adatti al proprio racconto, specie se a doverlo fare sono delle ragazze della scuola media. Inoltre, anche per qualcuno che ha già programmato, la cosa non è assolutamente immediata, poiché è necessario conoscere bene l'interfaccia di Alice e tutte le particolarità relative alle sottoparti degli oggetti, ed effettuare numerosi tentativi prima di ottenere l'animazione corretta.

Per comprendere meglio tale concetto è significativo elencare i metodi presenti in Alice 2 e comuni a tutti gli oggetti (senza darne una descrizione dettagliata poiché molti sono stati descritti nelle sezioni precedenti ed altri sono di facile intuizione), per poi vedere con quali metodi sono stati sostituiti in Storytelling Alice.

I metodi relativi ad Alice 2 sono i seguenti:
object.move(direction, amount), object.turn(direction, amount), object.roll(direction, amount), object.resize(amount), object.say(message), object.think(message), object.playsound(sound), object.move to(target), object.move toward(amount, target), object.move away from(amount, target), object.orient to(target), object.point at (target), object.turn to face(target), object.set point of view to(target), object.set pose(pose), object.stand up(), object.move at speed(direction, speed) object.turn at speed(direction, speed), object.roll at speed(direction, speed), object.constrain to point at (target), object.constrain to face(target).

Di seguito, come esempio, sono elencati alcuni dei metodi implementati in Storytelling Alice, pensati non soltanto ispirandosi ad azioni tipiche per un racconto, ma anche seguendo i possibili gusti di ragazze della scuola media. Essi sono specifici per determinate categorie di oggetti, poiché persone, animali e cose non possono compiere le stesse azioni, e per alcune categorie i metodi sono gli stessi di Alice 2. Inoltre per alcuni specifici personaggi sono stati scritti dei metodi particolari che non sono comuni a tutta la categoria ma solo ad essi. Per quanto riguarda tali metodi è anche possibile visualizzarne e modificarne il codice, sia in Storytelling Alice che in Alice 2 (infatti anche in Alice 2 erano presenti dei metodi specifici per alcuni oggetti, anche se in numero minore).

Metodi relativi alle persone:

Say: <person> say <string>

Think: <character> think <string>

Play sound: <person> play sound <sound>

Walk to: <person> walk to <target>

Walk offscreen: <person> walk offscreen

Walk: <person> walk <distance in meters>

Move: <person> move <direction> <distance>

Sit on: <person> sit on <target>

Lie down: <person> lie down on <target>

Kneel: <person> kneel

Fall down: <person> fall down

Stand up: <person> stand up

Look at: <person> look at <target>

Look: <person> look <direction>

Straighten up: <person> straighten up

Turn to face: <person> turn to face <target>

Turn away from: <person> turn away from <target>

Turn: <person> turn <direction> <amount>

Touch: <person> touch <target>

Keep touching: <person> keep touching <target>

Metodi relative ad altri personaggi (personaggi fantastici o non umani come gli animali):

Say: <character> say <string>

Think: <character> think <string>

Play sound: <character> play sound <sound>

Move to: <character> move to <amount> <direction> <target>

Move: <character> move <direction> <distance>

Look at: <character> look at <target>

Look: <character> look <direction>

Stand up: <character> stand up

Straighten up: <character> straighten up

Turn to face: <character> turn to face <target>

Turn away from: <character> turn away from <target>

Turn: <character> turn <direction> <amount>

Roll: <character> roll <direction> <amount>

Metodi relative agli oggetti e metodi specifici per la telecamera:

Turn: <object> turn <direction> <amount>

Roll: <object> roll <direction> <amount>

Straighten up: <object> straighten up

Move: <object> move <direction> <amount>

Resize: <object> resize <amount>

Move to scene tripod: <camera> move to scene tripod <scene tripod>

Get close up of: <camera> get close up of <target>

Get two shot of: <camera> get two shot of <target1> and <target2>

Get character’s view: <camera> get character’s view <character>

Show subtitle: <camera> show subtitle <string>

Show title: <camera> show title <string>

Fade to black: <camera> fade to black

Fade up from black: <camera> fade up from black

Point at: <camera> point at <target>

Move to: <camera> move to <distance> <amount> <target>

Move: <camera> move <distance> <amount>

Turn to face: <camera> turn to face <target>

Turn away from: <camera> turn away from <target>

Turn: <camera> turn <direction> <amount>

Stand up: <camera> stand up
Roll: <camera> roll <direction> <amount>

Tra le cose che sono state aggiunte vi sono anche due pulsanti: “add new object” nell’albero degli oggetti (mostrato in figura 5.1), che consente di aprire la galleria per aggiungere un oggetto, e “done adding objects” che si trova nella finestra per l’inserimento degli oggetti (nell’albero degli oggetti) e serve per tornare nella schermata principale. Sebbene pulsanti analoghi fossero già presenti nell’interfaccia, si è pensato di aggiungerli, valutando le difficoltà che avevano le ragazze nel trovare quelli già integrati in Alice 2. Tale aggiunta ha quindi solo dei vantaggi dal punto di vista pratico e non è fondamentale.

Sempre per semplificare l’uso dell’interfaccia, per ogni oggetto si è scelta una lista di metodi meno comuni, cioè che di solito non vengono utilizzati in un racconto, ed essi sono stati raggruppati nell’insieme “seldom used methods”. Quindi nel pannello relativo ai metodi viene mostrata prima la lista di metodi più comuni, e, se l’utente lo desidera, anche la lista di quelli utilizzati più raramente, semplicemente cliccandoci sopra. In tal modo si evita di disorientare le ragazze che potrebbero far confusione e non riuscire a trovare il metodo più appropriato.

È stato anche modificato il modo in cui vengono inseriti gli oggetti automaticamente: non più al centro del mondo (poiché, nel caso in cui l’utente abbia mosso la telecamera, l’oggetto potrebbe non essere più visibile) ma in una locazione dipendente dalla posizione della telecamera, in modo tale che la maggior parte degli oggetti inseriti risulti visibile.

Con il proposito di migliorare le storie che possono essere create per attirare maggiormente l’interesse delle ragazze, è stata apportata quella che probabilmente è una delle modifiche più significative del programma: è stata infatti abilitata la

creazione di scene multiple. È facile capire come per un racconto lungo e ben strutturato sia necessaria più di una scena, possibilmente con dei cambiamenti di luogo. In Storytelling Alice è possibile creare più scene premendo il pulsante “create new scene” mostrato in figura 5.1. È quindi anche possibile passare da una scena all’altra sia in fase di costruzione della scena stessa, sia in fase di esecuzione, avendo prima impostato ciò con le adeguate istruzioni nel programma.

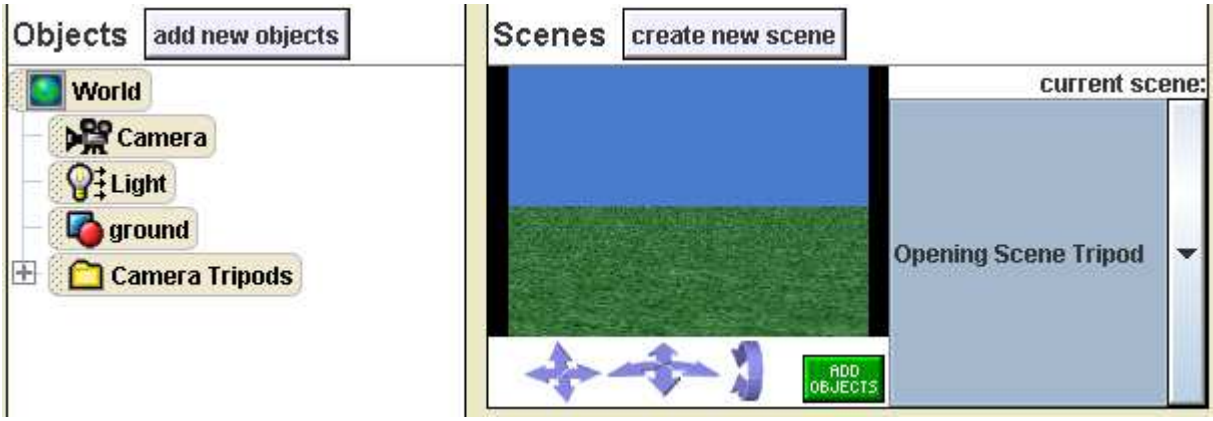


Figura 5.1: Cambiamenti nell’interfaccia di Alice

Inoltre, sempre con lo scopo di motivare le ragazze nel loro lavoro, sono stati aggiunti numerosi nuovi personaggi nella galleria degli oggetti 3D, più adatti ad essere utilizzati nelle storie immaginabili dalle studentesse. Essa è stata poi modificata cambiando le categorie di oggetti per facilitarne la scelta e la ricerca.

Infine, l’ultima modifica significativa apportata è stata quella dei tutorial: sono stati modificati quelli già presenti in Alice 2 e ne sono stati aggiunti di nuovi, specifici per gli obiettivi da raggiungere. In particolare viene spiegato in maniera più dettagliata come creare un racconto completo ed utilizzare l’interfaccia modificata per creare e gestire nuove scene.

6 CONCLUSIONI

Dall'analisi effettuata, tenendo anche in considerazione il discreto successo che il software ha ottenuto nelle università americane in cui è stato introdotto, si intuiscono le grandi potenzialità che esso presenta dal punto di vista didattico. Ma occorre prestare attenzione: proprio per la sua semplicità, alcuni suoi punti di forza potrebbero essere visti come degli svantaggi. Infatti, non permettendo di commettere errori di sintassi, lo studente non si abitua a capire dove sbaglia e a correggere i suoi errori. Inoltre, proprio il comodo ambiente "drag and drop" e l'interfaccia semplice ed intuitiva, potrebbero non far rendere conto allo studente di cosa significhi effettivamente programmare. Quindi, anche basandomi sulla mia esperienza personale, ritengo che l'insegnamento della programmazione nel modo classico non possa assolutamente essere sostituito, per quanto frustrante e complicato possa essere. Tuttavia, il software Alice potrebbe comunque essere utilizzato per l'introduzione ad un corso di programmazione, in quanto agevola l'apprendimento dei concetti teorici dei linguaggi di programmazione orientati agli oggetti e fornisce una prima esperienza dal punto di vista pratico. Sta quindi nell'abilità del docente far comprendere il parallelo con linguaggi come Java, in modo che gli studenti capiscano che programmare nella maniera classica non è semplice come programmare in Alice. Inoltre ritengo fondamentale l'uso di un testo come quello consultato durante la stesura di questa tesi, poiché uno studente alle prime armi potrebbe non collegare correttamente il suo operato con i concetti teorici della programmazione o potrebbe utilizzare il software in maniera errata. Per questo credo che Alice 3 non abbia ancora tutti i requisiti per essere introdotto in un ambiente universitario, cosa che potrebbe però avvenire in futuro, non appena verrà scritto un adeguato libro di testo. Invece per quanto riguarda Alice 2, per il momento potrebbe essere utilizzato in un contesto universitario (come è già stato fatto), per poi sostituirlo in futuro con Alice 3 ed introdurne l'uso nella scuola media, dove potrebbe dare un fondamentale contributo nel motivare gli studenti ad apprendere una scienza importante come l'informatica.

7 BIBLIOGRAFIA

Per l'analisi del software Alice è stato utilizzato il materiale che si trova sul sito ufficiale di Alice, ed inoltre è stata molto utile la consultazione del libro di testo di Alice 2. Tutto il materiale consultato è stato trovato a partire dai seguenti siti:

[1] http://en.wikipedia.org/wiki/Alice_(software)

[2] http://www.alice.org/index.php

[3] http://www.aliceprogramming.net

[4] http://www.dukechronicle.com/articles/2011/06/23/alice-project-introduce-children-computer-science

[5] http://www.cs.duke.edu/csed/alice/

Il libro di testo utilizzato è invece:

[6] Learning to Program with Alice, 2nd by Wanda P. Dann, Stephen Cooper, Randy Pausch