Exploring

Alice

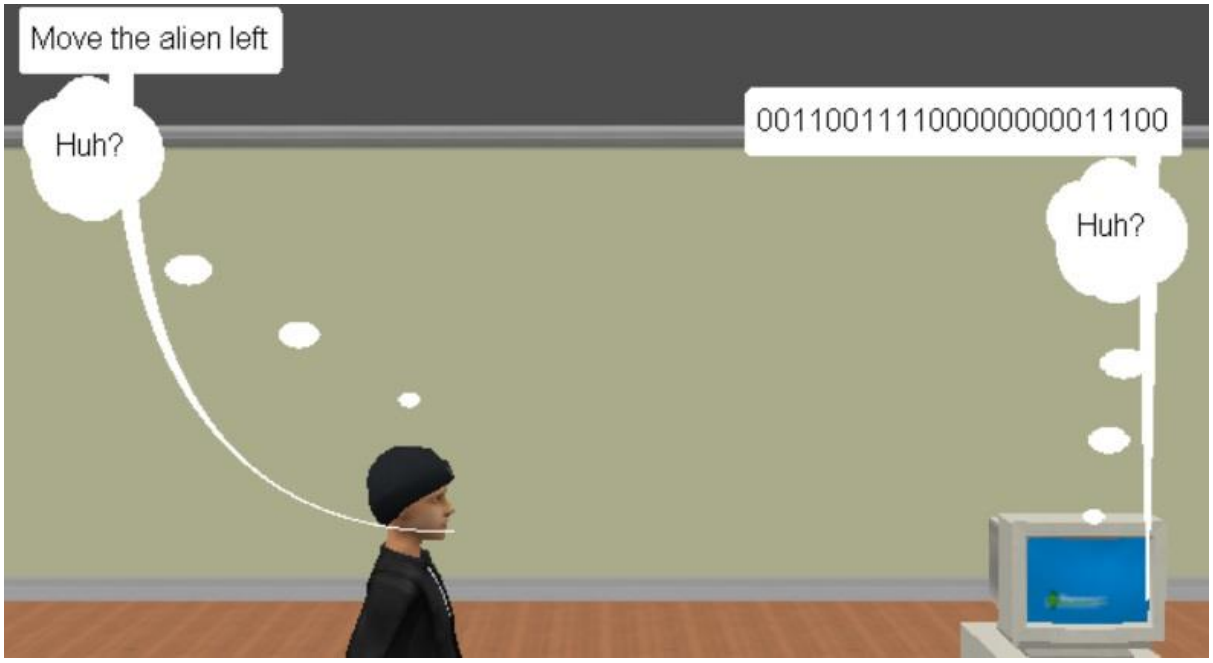
Contents

1 Overview

1.1 What is Alice?

Alice is a programming language. Ok, so what's a programming language?

Computers only think in electronic circuits that can be on or off. On is seen as '1', and off is seen as '0'. So trying to communicate with a computer directly could work something like this:



A programming language, such as Alice, can be loaded onto the computer so that we can give instructions in Alice-language (which is easy to learn), and it translates these instructions into the 0s and 1s that the computer understands.

Other programming languages include Java and C. Alice is, in fact, based on Java.

Alice was designed as a learning tool, to make learning how to program easy and fun. It's a great way to create animated stories and games, let your imagination run wild, and at the same time learn programming skills that many used to find really difficult.

Because it's a learning tool, it's not as powerful or as flexible as Java, and those who find they really enjoy programming may want to learn Java or C later.

Alice allows you to create a 'world', add some people, animals and scenery, and make your characters do something. Alice worlds are 3D (three-dimensional) – this means that everything has height, width and depth, and can move forward and back, up and down, or left and right.

Below is a world created with Alice.

Alice can be used to write animated stories or cartoons, stories that interact with the user, and games. In this introduction course, we'll only be looking at very simple animated stories. In the full course, we'll look at more complicated animation, and interacting with the user.

1.2 Some programming words

If you find you enjoy programming, and later want to go further with it, it will help if you start using the right words for programming-related things from the beginning. Otherwise it will become confusing later. Here are some words we'll be using in tis course, and what they mean to programmers.

Object - A thing that's a part of your program. For example, in a game, an object could be an alien, a child an aeroplane or a tree.

Property - Something that an object has. For example, an alien may have a colour and a size; it also may have a head, a hand etc.

Method - A way of describing how something is done. There are two kinds of methods: procedures and functions. The difference will become clear later.

Class - A blueprint for creating similar objects.

Event - Something that happens. For example, the user clicks the mouse or presses a key on the keyboard.

Code - The instructions that make up a method.

2 Creating a Scene

Developing an Alice world consists of two tasks: creating a scene, and adding some action. Alice has a scene editor for the first task, and a code editor for the second. You can think of it as staging your own theatre production or movie. In this lesson, we'll look at how to create a scene.

The program will have:

A Scene: This is the world you are building, and in Alice, it's referred to as 'this'.

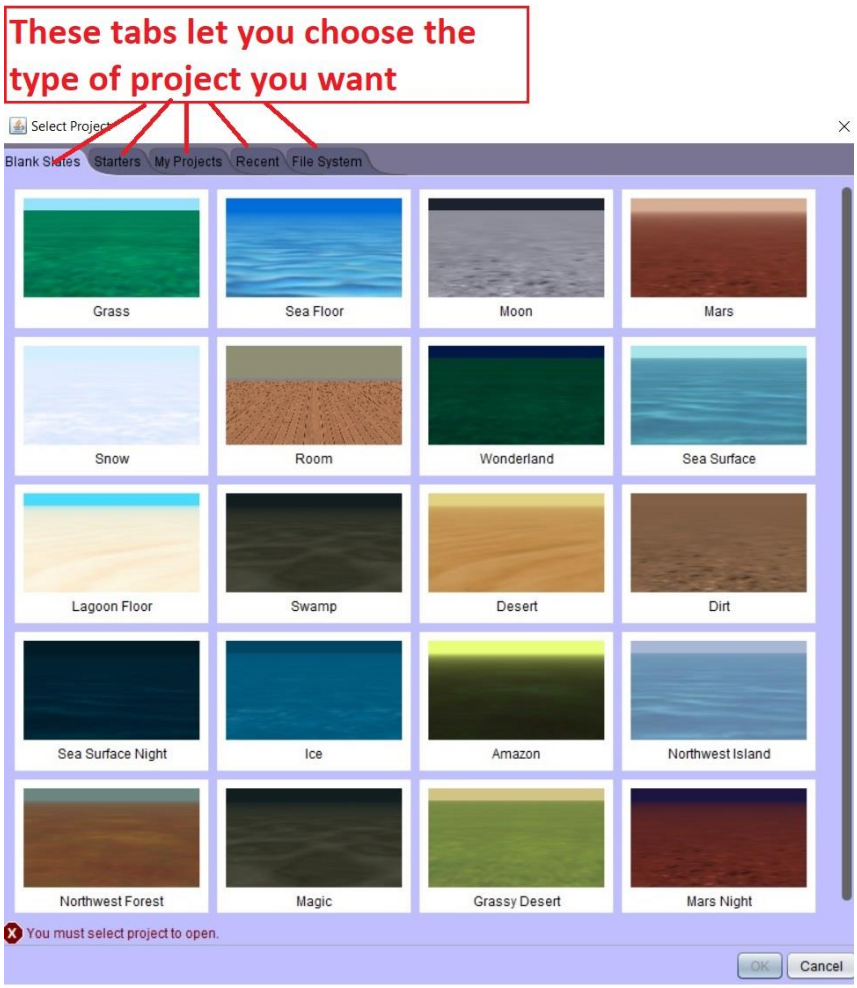
The Ground: This belongs to the scene. You can choose different types of ground, for example grass, desert, or moonscape.

A camera: There is only one camera, and it is the 'eye' that people will use to look into your world. The camera can be moved around as needed. It belongs to the scene.

Some objects: Alice has a gallery of objects that you can add to your scene, including people, animals, monsters, scenery and props.

2.1 The scene Editor

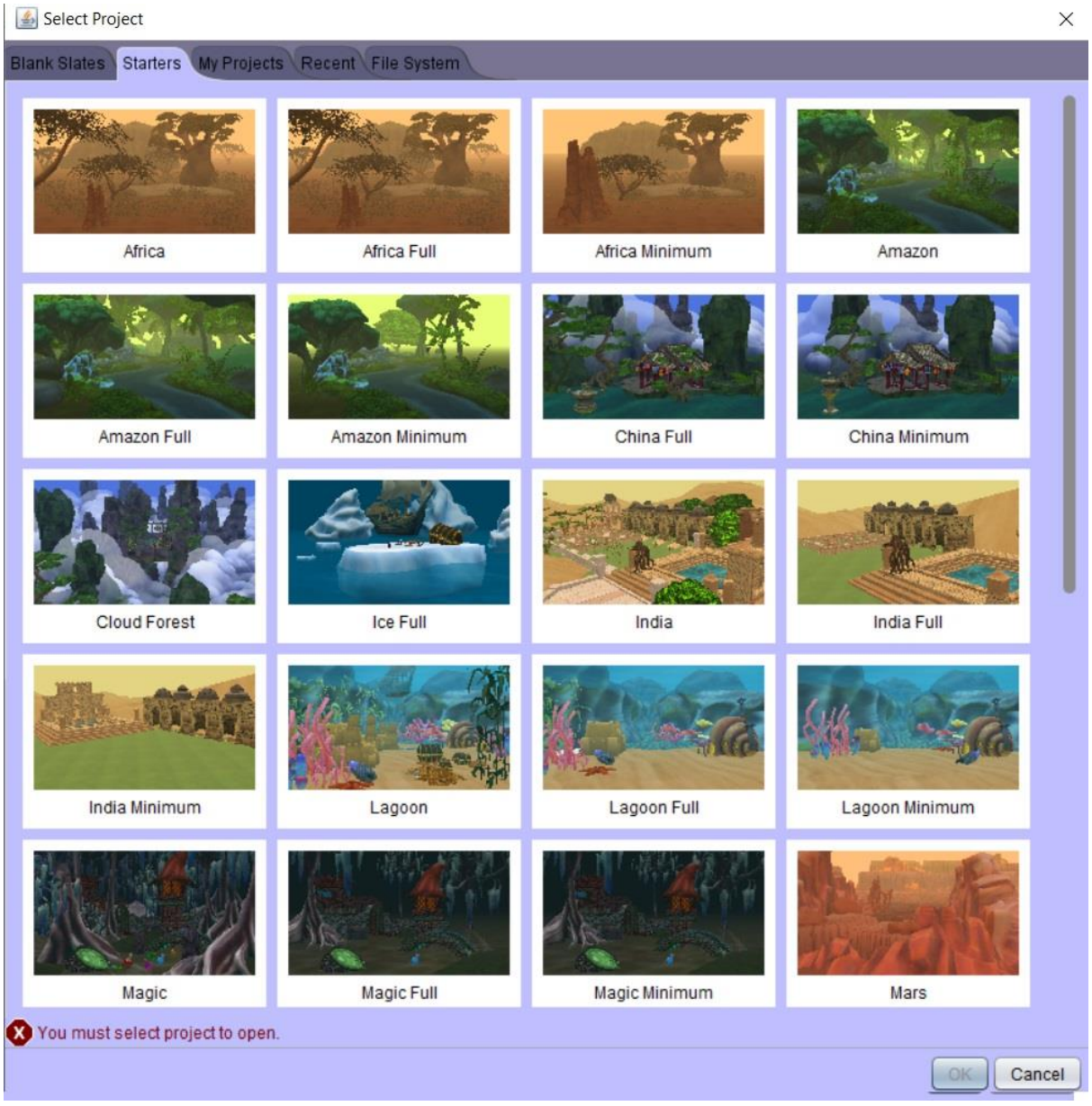
When the Alice program starts, it asks you to choose a project. A project is an Alice world: it may be one that you've created, or a new blank project, or a 'starter' project, where some of the world building has already been done for you. The screen you first see will look like this.



The tabs at the top let you choose what kind of project you want to work with. You would use one of the first two (Blank Slates and Starter Projects) when you're starting a new world. You'd use one of the other tabs if you want to open a world that's already been saved.

In the picture above, the Blank Slates tab is highlighted, and the rest of the screen shows different types of worlds to choose from. Blank Slates give you an empty world, with ground and atmosphere settings to suit different types of scenes. If you want a Blank Slate, pick the one that best suits your game or story, and click OK.

If you'd rather have a world that's already had some scenery and props added, click the 'Starters' tab. You will then see a screen that looks like this:



As you will see, there are a lot of exciting worlds to choose from. Use the scroll bar to have a look at what's available, then choose the one that best suits your story.

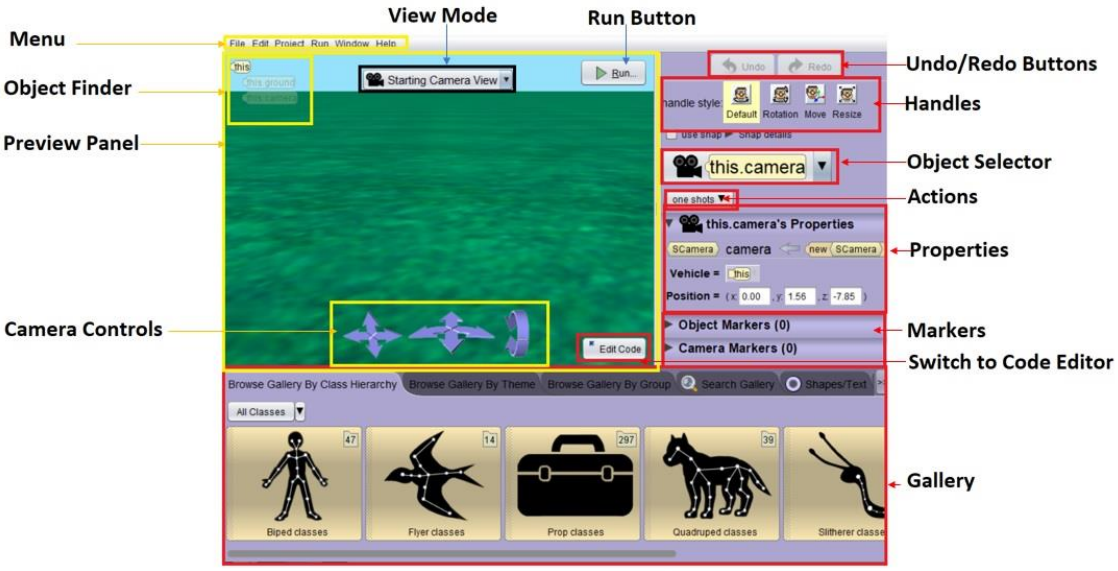
For most of the course, we'll be using blank slates, because they're easier to work with when you're learning. Later, when you're more confident, you can experiment with the different starter worlds to create beautiful games and stories easily.

Once you've chosen your world, you will see this screen:

As we mentioned earlier, Alice has two editors: a scene editor and a code editor. The screen above, which is the first screen you see, is the code editor. We'll learn how to work with this later.

To switch to the scene editor, click the button marked 'Setup Scene'.

The screen will now look like this:



This screen has a lot of different parts to it, so while you're learning Alice, it's a good idea to have this picture nearby to help you find things.

For now, we'll just talk about a few of these things.

Menu: You'll mostly use this to save your project, or open a different one.

Preview Panel: This is where you can see what your world looks like as you build it.

Camera Controls: These let you move the camera to different viewpoints of your world.

Object Finder: This shows a list of all the objects in your world. In a brand-new world, there will only be three: 'this' (your world), this.ground (the ground) and this.camera (the camera). When you add new objects to your world, they will also be listed here. In programming terms, 'this.' before an object name means the object belongs to 'this' – so these objects belong to the world.

Gallery: These are all the objects you can choose to add to your world.

Undo and Redo: Used to undo or redo the last thing you did. You can also use the shortcuts CTRL-Z (undo) and CTRL-Y (redo).

2.2 Saving your world

It's very important to save your world often, so you don't lose all your hard work if the computer crashes or if you make a mistake that messes up your world. When you open a new project, you should always save it immediately as soon as the scene editor appears on the screen. To do this, click File and then Save. Give your project a name.

Then save often as you're working, perhaps every 5 or 10 minutes. To save time, CTRL S pressed together does an instant save.

It's a good idea, as your world grows and becomes more complicated, to save under different names as you go along, so you can go back to an earlier version if you find you don't like some of the changes you've made to it. For example, you could name it SnowWorld1 the first time you save, then when you've made a lot of changes, save is as SnowWorld2, and so on. Use File, then Save As, to save under a different name.

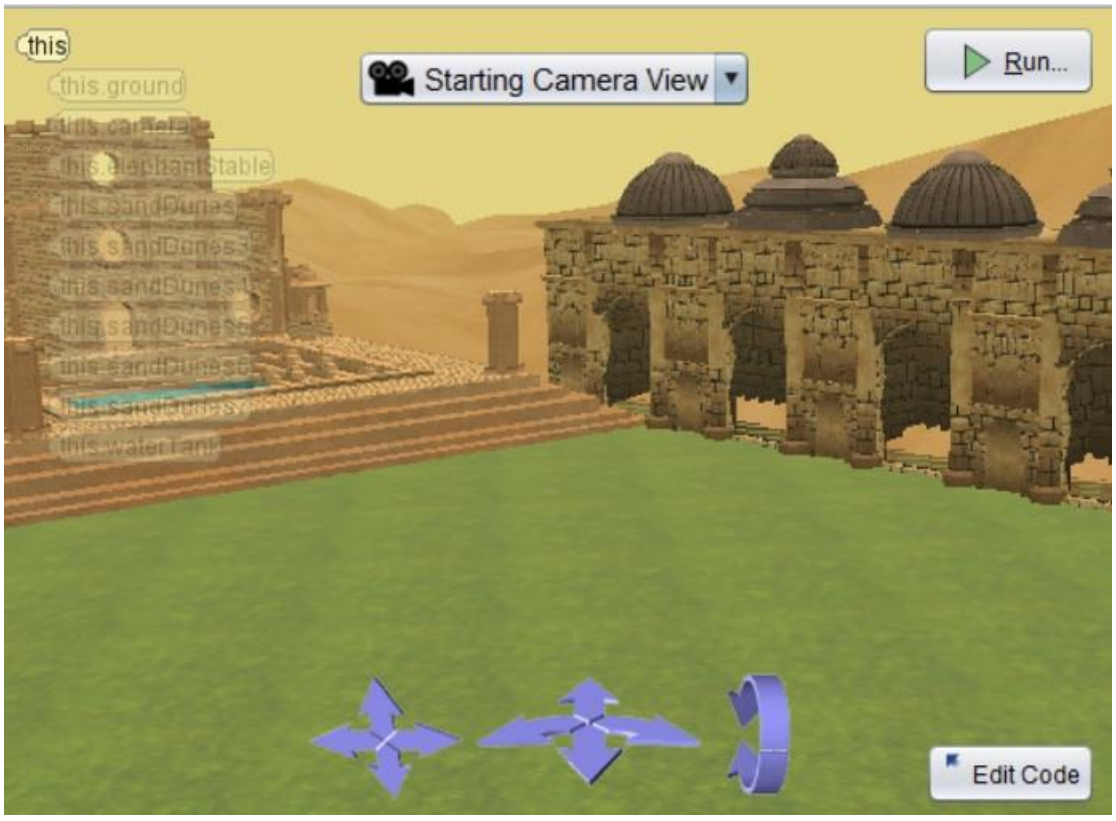
It's also a good idea, for worlds that you really don't want to lose, to have a copy of them somewhere that's not on your computer, so if your computer disk becomes faulty, you haven't lost it. You could save to a flash drive, or, if you're connected to OneDrive or DropBox, save it there. If you don't have any of these, you could perhaps email it to a friend for safekeeping. Your projects can be found under your computer's Documents folder, in Alice3\MyProjects.

2.3 The camera

The next thing we'll look at is the camera. To practice using the camera, it's best to open a world that already has something in it, so practice with one of the starter projects. IndiaMinimum is a nice one to play with, as it has enough in it to get you started, but not so much that it becomes confusing.

Use File and New, click the Starters, and choose IndiaMinimum. Before you do anything, choose File – Save As to save it under a new name – it's not a good idea to make changes to the original starter projects. Call it MyIndiaWorld or something similar.

You'll see this screen:

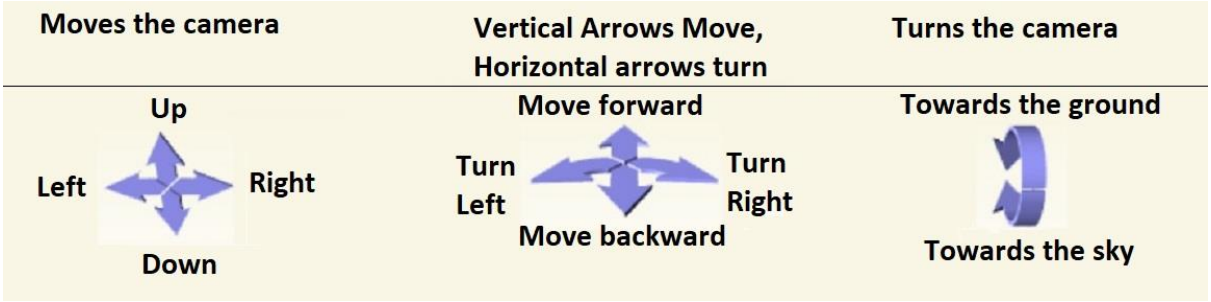


By adjusting the position and angle of the camera, you can move around the scene in 3D, just as if it was a real world. You can move the camera up and down, left and right, and forward and backward. You can also tilt it forward or backward to get a different angle, and also swivel it to the left or the right. Try to imagine this with a real camera before you start using the camera controls.

2.3.1 The camera control buttons

The camera control buttons are quite sensitive, so when you're starting, it's best to use single clicks to get small movements. When you're confident, you can hold down the mouse button to get bigger movements.

The buttons are at the bottom of the preview panel, and they work like this:



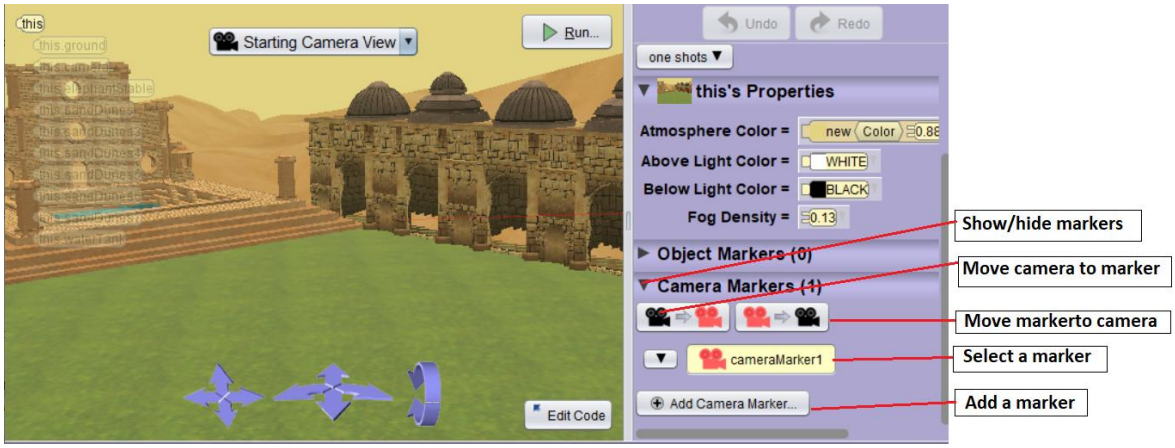
Remember, if you turn the camera left, it will look as if the scene is moving to the right.

If you move the cursor ahead of where the camera control is pointing while holding down the mouse button, it will move faster.

2.3.2 Camera Markers

As you will discover when you're experimenting with the camera, it's quite easy to get 'lost' in the virtual world. Camera markers store the position of the camera at a given point, and they make it easy to get back to your original viewpoint if you do get lost. They're also useful when you write code to add action to your scene, because you can move the camera to preset locations as your story or game moves along.

The camera marker controls are below the properties panel in the scene editor. If you can't see them, use the scroll bar on the left to bring them into view. They look like this:

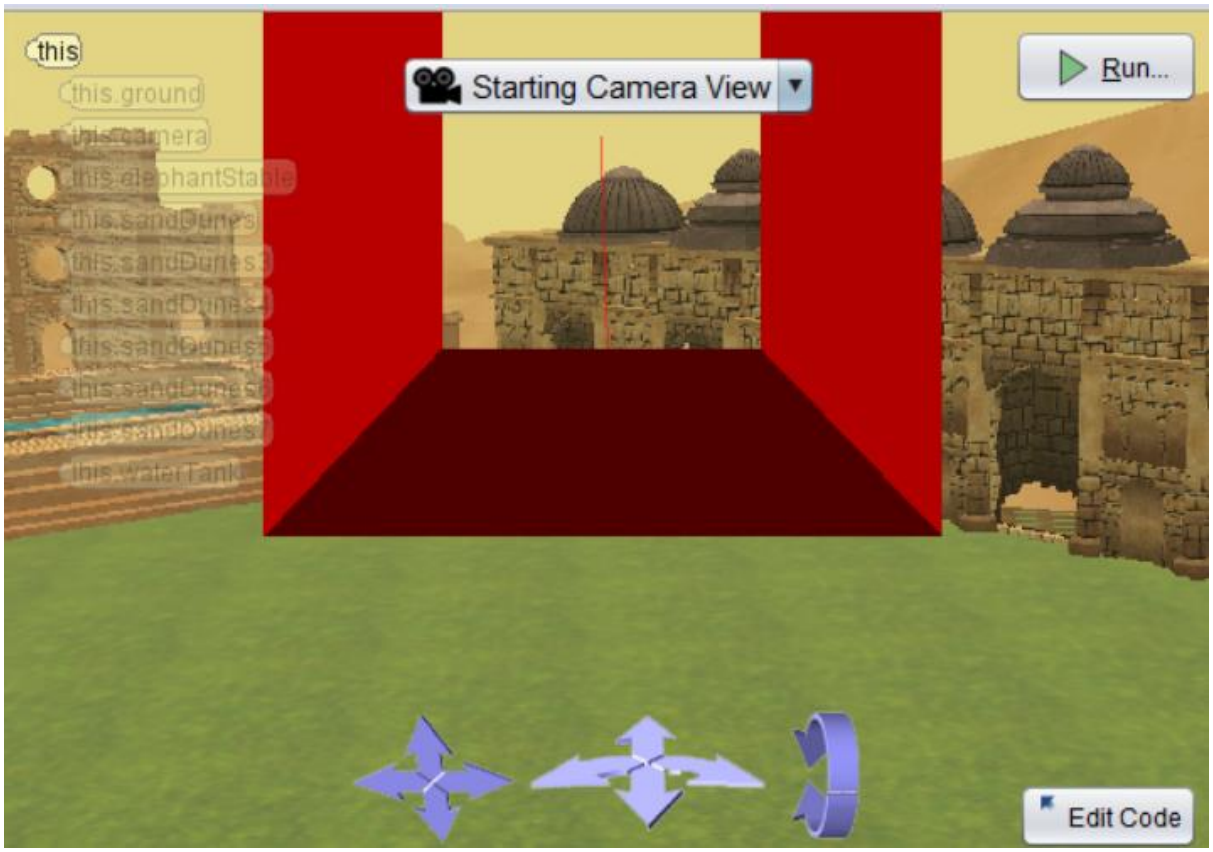


To set a new camera marker at the camera's current position, click 'Add camera marker', and give a name to the marker.

To move the camera to an existing camera marker, click the name of the marker, then click the button marked as 'Move camera to marker' in the image above.

To move a marker to a different position, first move the camera to the new position, then click the name of the marker, then click the button marked 'Move marker to camera' in the image above.

If you move the camera behind a camera marker, the camera marker shows in your scene like this:

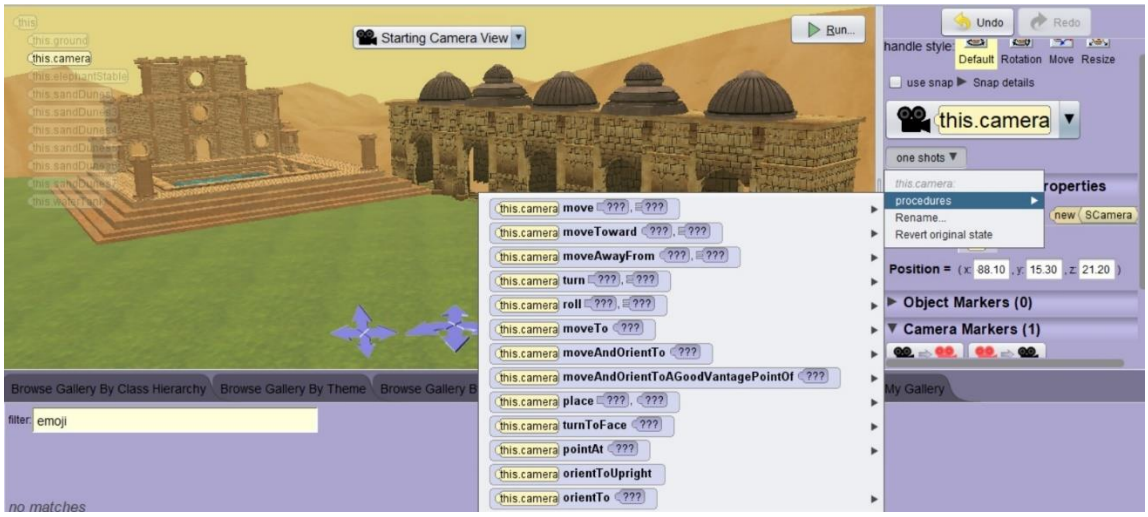


When you run the program, you won't see it; it's just there in the scene editor to show where the marker is.

It's always a good idea when starting a new project to immediately set a camera marker, so you can always go back to your starting position.

2.3.3 Using one shots to control the camera

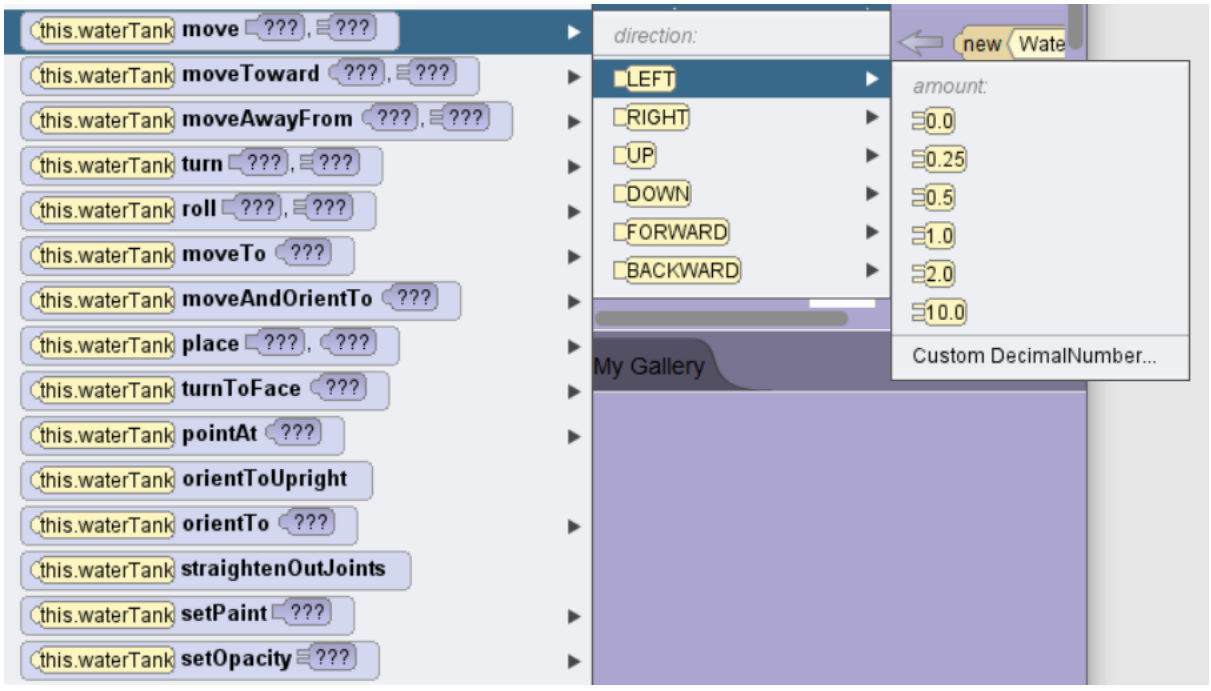
One shots are actions (procedures) that you can ask an object to do while you are setting the scene. Many of them are the same as procedures that you can use when you're adding action to the program, but the difference is that one shots only happen once when you request them in the scene editor. They give you more control over objects during scene setting. Using one shots, you can ask the camera to move forward or back, turn, move to a good vantage point of an object, and many more actions.



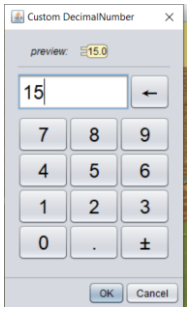
To use one shots for the camera, first select the camera as the object you want to work with. There are two ways of doing this. You can click 'camera' on the list of objects at the left of the preview panel, or you can click the arrow on the object finder (just below the handles at the top right), and choose the camera from the list of objects it will show you.

Next, click One Shots (just below this), then click 'procedures'. It will show you a list of things you can ask the camera to do. Choose the one you want.

In most cases, it will need more information about what you want to do. If you chose 'move', you would see something like this.



You first need to tell Alice which direction the camera must move, then tell it how much it must move by. If you want to move by an amount not shown in the list, choose 'Custom Decimal Number', and a pop-up will allow you to type in the number you want, like this:



One shots can often be a much quicker way of controlling the camera than the camera controls, but it needs a bit of practice to use them easily.

2.3.4 Controlling the camera using its properties

Another very quick way to move the camera a long distance is to change its properties.

The properties panel is directly below One Shots. Here, you can change the camera's position by changing the x, y and z settings. x, y, and z contain numbers that represent a position in three dimensions: width, height and depth. You can experiment with changing them. Remember that any movement depends on which way the camera is facing.

2.4 Objects, classes and the gallery

The Gallery

The Gallery at the bottom of the scene editor allows you to choose the type of object you want in your scene. The gallery contains classes, which are blueprints from which objects are made. The class defines what the object looks like, its properties and what it can do. The tabs at the top let you choose how you will search for a suitable object. The choices we'll look at on this course are:

Browse by class hierarchy: Classes define what properties an object has (eg size, position, colour). Some objects have arms and legs, others don't. Classes also define the object's methods, in other words what it can do (eg move, turn). Many classes have some of the same properties and methods in common. So these common methods and properties can be defined in a **superclass**, and all **sub-classes** inherit these methods and properties. Examples of superclasses are 'biped' (a thing with two legs) and 'quadruped' (a thing with four legs.) All bipeds have two legs and two arms, and are able to move them. They all have

similar joints, and they are all able to 'say' and 'think'. In this part of the gallery, classes are arranged by class hierarchy – superclasses split into subclasses.

Browse by theme: This part of the gallery is arranged by theme, so that it's easy to find objects that will be needed for they type of world you are building. Themes include Magic, Ocean and Outer Space.

Browse by group: Groups include animals, plants, people and household items. It can be a quick way to find what you need.

Search: This lets you type in a keyword to search for the type of object you want, eg Tree.

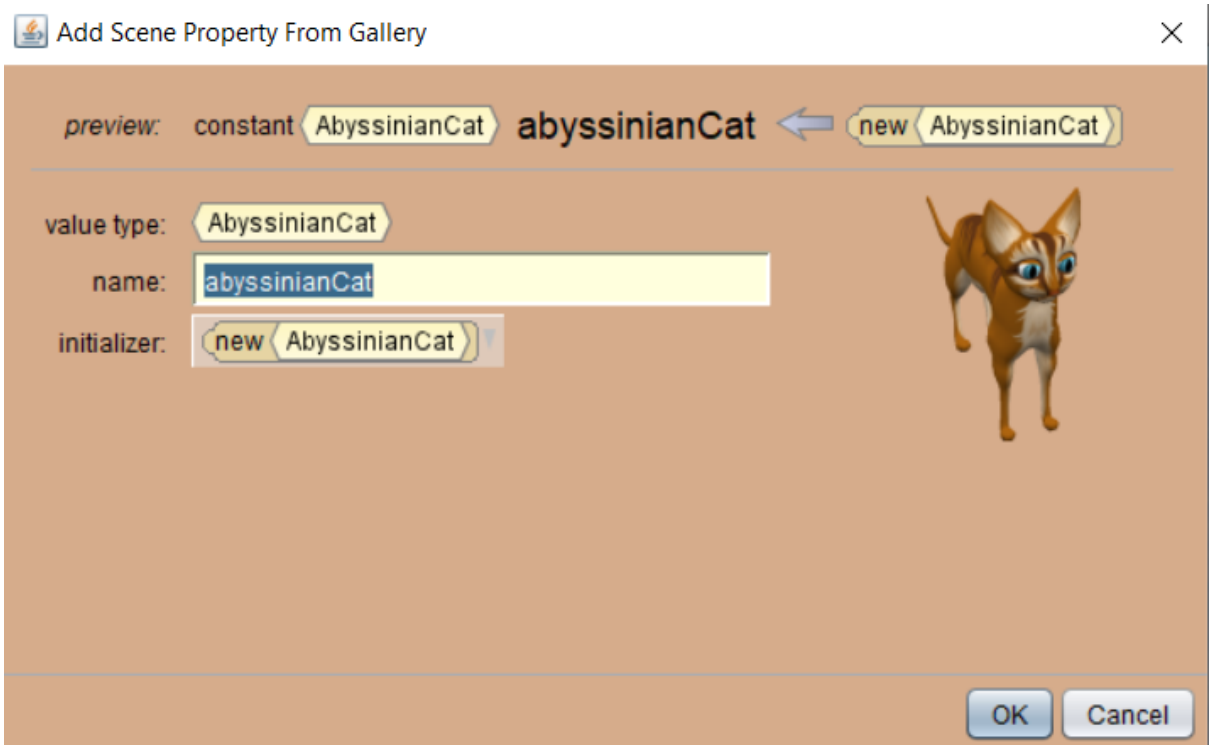
Shapes/Text: This contains a few useful shapes and text boxes.

2.5 Adding an Object

Let's open a new project with a blank slate using the Snow background. Set a camera marker so you don't get 'lost' later. When learning to add and position objects, it's much easier to start with a blank slate, because the starter projects already have a lot of objects in them, and it's easy to 'lose' your new object amongst all the scenery. Browse the gallery and find an object you would like to add to the scene.

To add an object, you can either double-click it, or click and drag it. If you double-click to add, it will always place the object at position 0:0:0, even if the camera is not pointing at this location. This can make you think the object hasn't been added at all. The other problem with using the double-click is that if there is already an object at this location (or in front of it), the bigger object will hide the smaller object, again making you think the object is lost. It's best to use the click and drag method so you have control of where the object will go.

A pop-up box will appear like this:



This lets you give your new object a name. Use a name that you will find easy to remember, and that fits in with your story or game. Names can only be made up of letters and numbers – no spaces or punctuation marks. A popular programming standard is to name things in 'camel case', where the first word begins with lower case, and all other words in the name begin with upper case eg fredSmith, scaryOgre. It's a good idea to get into the habit of doing this, in case you get serious about programming later.

Once you've typed in the name and clicked OK, you will be able to see the object in your preview panel. You'll notice that the object's name now appears in the list of objects. The object has handles around it, to show that it's selected and able to be moved.

We'll learn how to move and resize the object in the next section.

Adding people is a bit more complicated, because you will need to choose what they look like, what they are wearing etc.

To add a person, browse the gallery by class hierarchy, and choose the biped class. You will see that there are classes available for different ages of people. Let's choose a Teen – click and drag the teen onto your screen.

You'll get a pop-up screen where you can make choices about the new person.

First choose near the top whether you want the person to be female or male. Next choose the skin colour below this. You can choose from the colours provided, or get adventurous and click Custom Colour to play with your own colour choice. You may want your character to be green to fit in with your story or game!

Next choose what they are wearing, either from the 'outfits' tab, or from the 'top/bottom' tab, where you can mix and match.

Then click the 'hair/hat' tab to choose from different styles. Lastly, click the 'face' tab to choose the shape of the face and the eye colour.

When you're happy with the way your person looks, click 'OK, then give them a name.

All objects under Alice are 3-D, and have height, depth and width. They also have **orientation** – in other words, which way are they facing in all dimensions. If you move an object forward, it will move in its own forward direction, which may not always be the same as moving forward on the screen.

They also have pivot points – the part of their body that will stay in the same place if they are turned. This can be different for different classes of object.

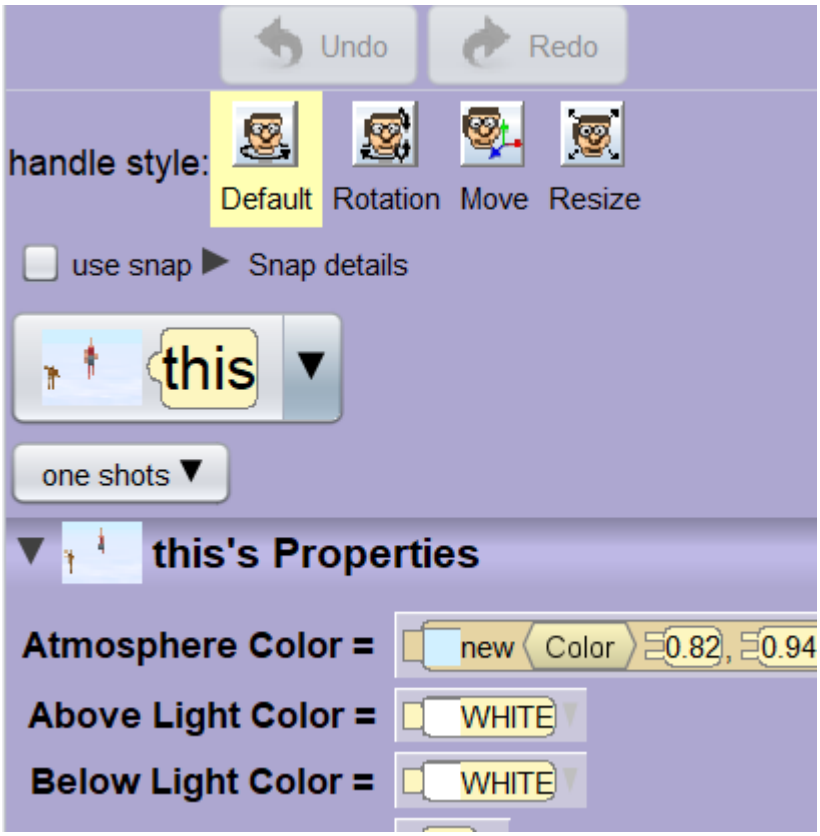
2.6 Moving, rotating and resizing an object

Objects can be moved and resized using the mouse. To do this, Alice has four different types of 'handles', which we'll look at below.

Objects can also be moved and resized using either Properties or One Shots, as we'll see a bit later.

The object must first be selected, either by clicking on it, or by choosing it from the list of objects on the left.

You can select the handle style that you want to use by clicking it on the top right of the screen, just above the properties panel.



The default handle:

This is the handle style that's selected when you first create an object. It can be used to move the object forward and back, but not up and down. It can also be used to turn the object left or right . Initially, the handle will be shown like this:

This ring allows you to turn the object. To rotate, move the mouse until the circle changes its appearance – usually it gets more shiny. You can then drag the circle left or right to rotate the object. It takes a bit of practice, because if you try to drag it when the circle is NOT shiny, it deselects the object.

To move the object using this handle, click anywhere on the object and drag it. The handles will change to look like this:

It will move along the line of its orientation – the white arrow shows which way is forward, the red shows right, and the green shows up. Remember, different objects may be orientated differently. The place where the three arrows meet is the object's pivot point.

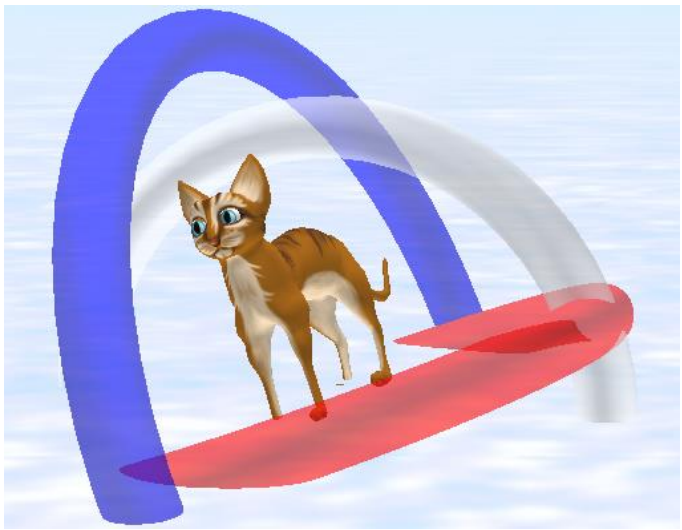
The rotation handle:

You would use this to rotate the object on any of the three dimensions, whereas the default handle only lets you rotate left and right. If you select an object when this handle style is selected, the handles look like this:

Use:

The blue handle to rotate forward or backward:

The white handle to rotate sideways:



The red handle to rotate left or right.

As with the default handle, you first have to move the mouse until the handle you want to use looks shiny.

The move handle:

This handle allows the object to be moved in all three dimensions: forward and back, left and right, and up and down. The handles look like this:

Drag the handle that points in the direction of move that you want.

The resize handle:

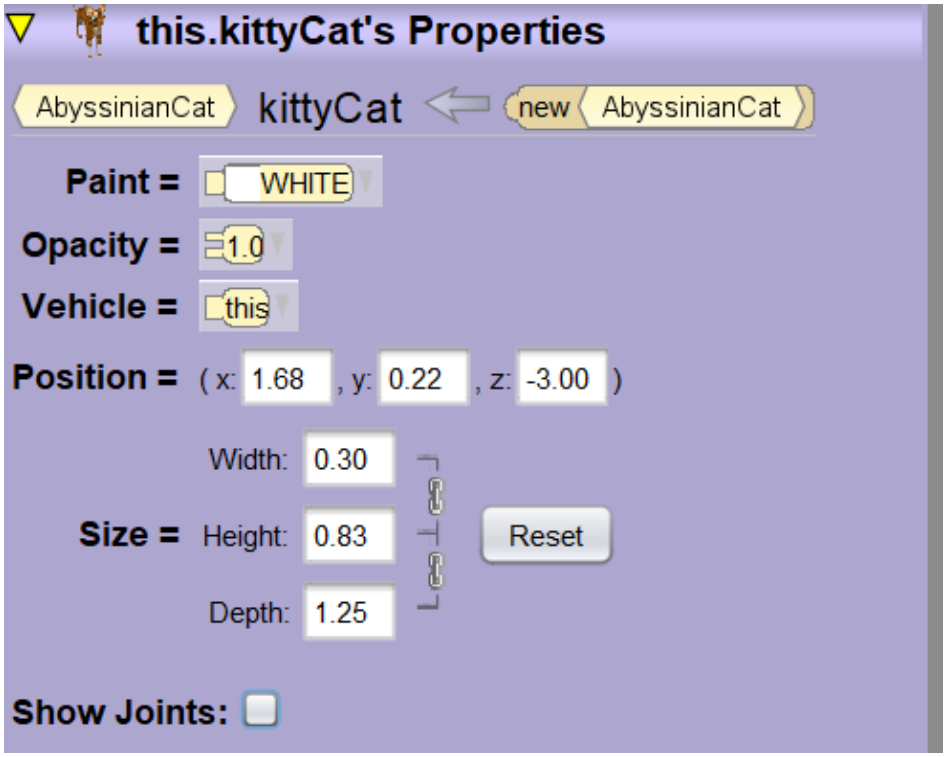
The handle looks like this:

Drag the arrow up to make the object bigger, or down to make it smaller.

2.7 Properties

The properties panel shows the properties of the selected object. Different classes of objects have different sets of properties, although most of them have several in common.

The properties panel for the cat looks like this:



You can change any of these properties.

Paint: Changes the main colour of the object

Opacity: Changes how see-through the object is. 0 is invisible, 1 is not see through at all, and numbers in between can be used to give a ghostly effect, like this:

Vehicle: You would use this if you wanted to 'attach' one object to another, so that when the second object moves or rotates, the first one also moves or rotates in the same direction. This can be useful if you want a man to sit in a boat, or on a horse, and move along with it. By default, the vehicle is set to 'this', which is the world.

Position: As with the camera, you can use this as a quick way to move an object a specific distance.

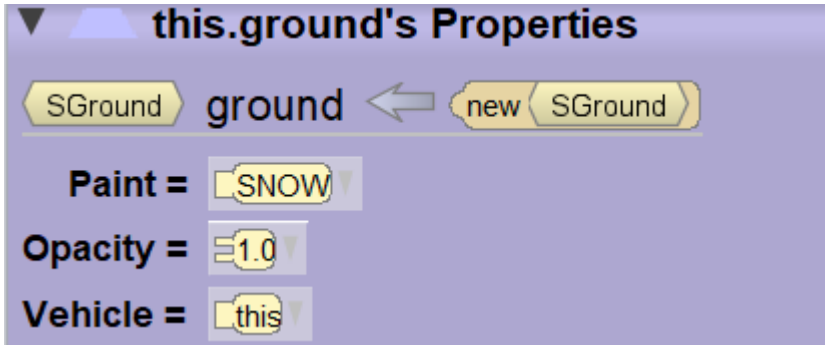
Size: This is a more accurate way of resizing the object. The objects always stay in proportion, so if you increase the width, the height and depth will automatically increase.

As well as being able to change the properties of the objects you create, you can change the properties of the scene (this), the ground, and the camera by selecting the one you want from the list of objects.

For the scene, the properties are:

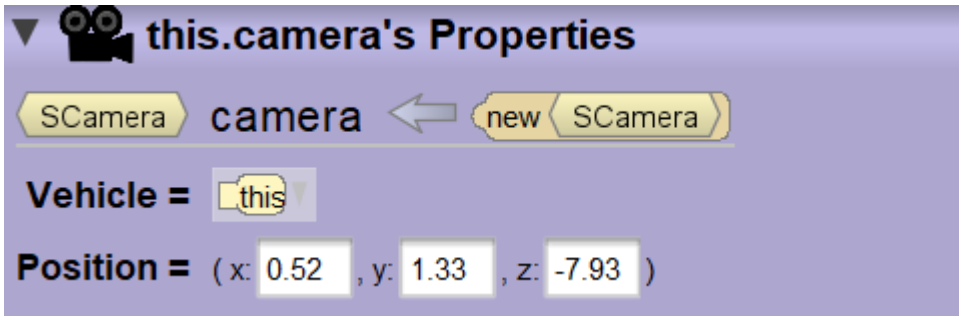
You can get special effects for your world by experimenting with these.

For the ground, the properties are:



Again, you can experiment with these to get the world you want.

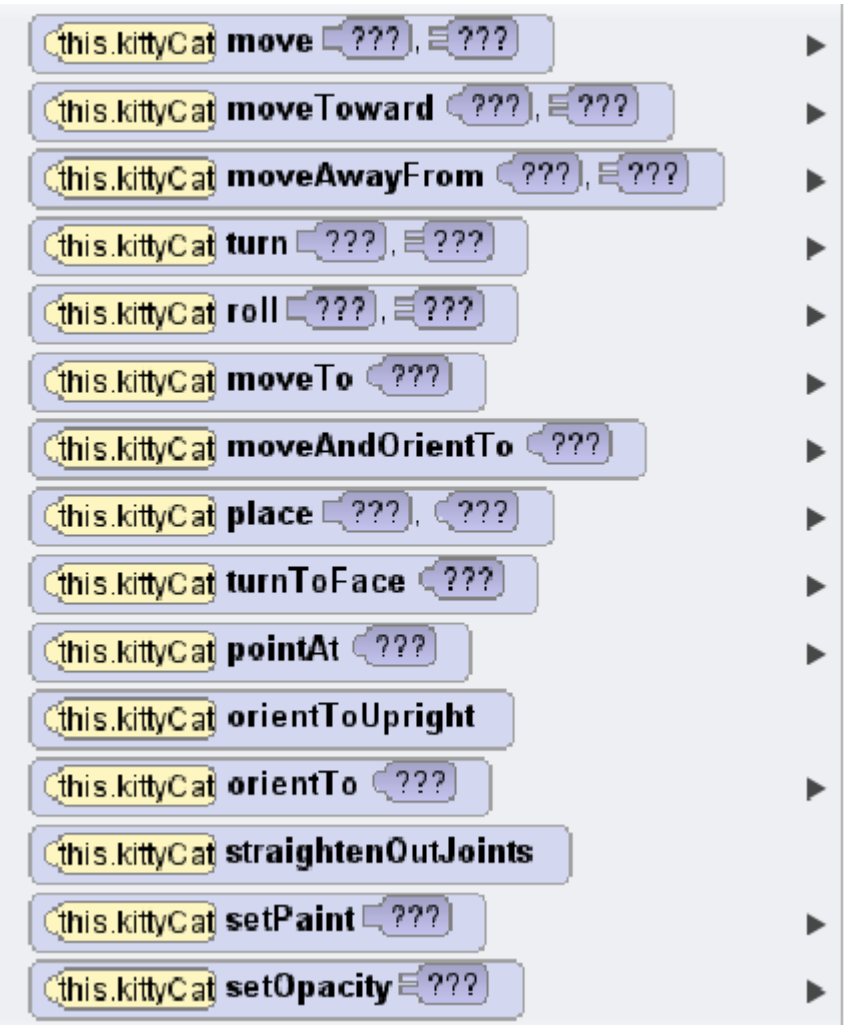
For the camera, the properties are:



The vehicle property becomes useful when you add action to your world, because you can use it to get the camera to follow a character.

2.8 One Shots

As with the camera, you can use One Shots to make the object do something while setting up the scene. This can be a more accurate way of turning or moving an object by a specific amount. Different classes of object have different procedures they can carry out. Clicking One Shots then Procedures when the cat is selected shows the pop-up below:



Important Note: When you use the 'move' One Shot to move an object, it moves it from the point of view of the object. So if you move an object left, it will move it to their left, which would be to your right if the object is facing you.

Please do Exercise 1 in the Practicals handout. Take your time to experiment and learn how to move, rotate and resize objects easily.

3 Adding some Action

To add action to an Alice world, you need to write a program. A program is a set of instructions, telling the computer step by step what to do and in what order. A computer can't think for itself; it will only do exactly what you tell it. So before you start, you need to be clear in your mind exactly what you want done. Then you will have to use 'Alice-language' to write an Alice program to describe the actions you want.

3.1 More programming words

Before we start, let's take a look at a few programming terms that we'll be using, and what they mean.

Method: A set of instructions for doing something.

Procedure: A type of method that is made up of a list of actions, e.g. moving a character or making it say something.

Function: A type of method that gets some information, e.g. get an aeroplane's height above the ground.

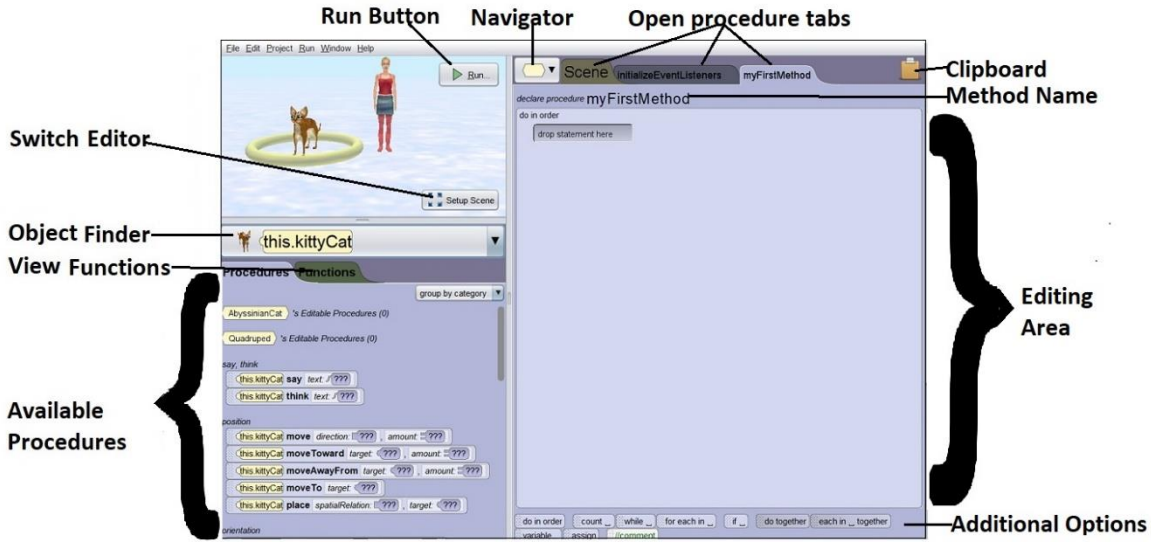
Parameter: The Cambridge dictionary gives the meaning of 'parameter' as 'a set of facts or a fixed limit that establishes or limits how something can or must happen or be done'. For example, a teacher may ask the class to write an essay of about 300 words on dogs. 'Dogs' and '300 words' are the parameters, because they describe what the teacher wants. In an Alice program, if we gave an instruction to move an aeroplane up by 3, the parameters are 'up' and '3'.

Object-oriented: This means that everything is centred around objects. All methods belong to an object.

3.2 The code editor

When you first open Alice, it will take you directly to the code editor. If you have been working in the scene editor, you would need to click 'Edit Code' to switch to the code editor.

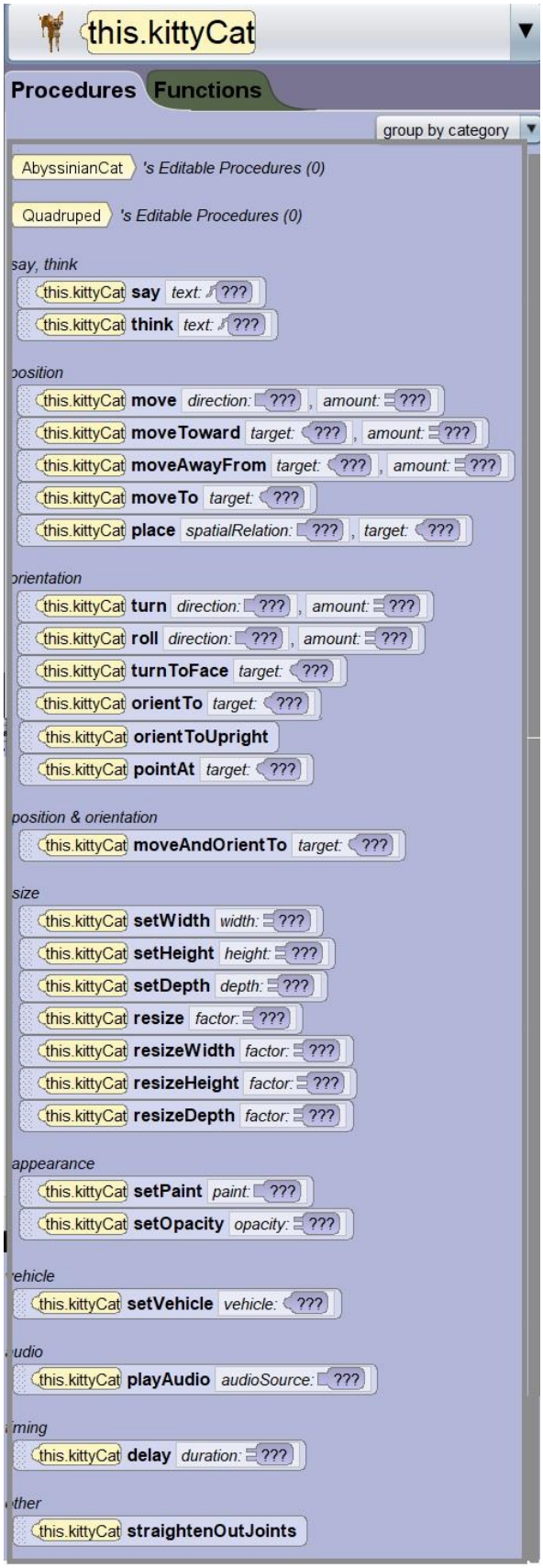
The code editor screen looks like this:



You're now ready to add some code into a procedure. Notice that the procedure tab myFirstMethod is highlighted, and the method name is also myFirstMethod. This method is called as soon as your new program is run, so the instructions for starting your program should always go there. In the next course, we'll learn how to write other procedures. For now, we will only work in myFirstMethod.

Alice classes already have a lot of built-in procedures and functions, and your Alice program will be made up of calling these. As we said earlier, procedures always belong to objects. We've already used some of these procedures when we used One Shots. Now we'll put together some of these procedures (in other words actions) into a program.

The first thing to decide for each step of the program is which object will be affected, and use the object finder to select this object. A list of the available procedures for that object will then be shown, like this:



You've already used a few of these with OneShots when you were setting up your scene. You can have fun experimenting with them to see what they do. You may have noticed that there are more procedures available when coding than there were in One Shots.

To add a procedure to your program, drag it from the available procedures into the editing area. Most procedures need one or more parameters, for example moveTo needs to know the target object that the object must be moved to. You will be asked to fill these in as soon as you've dragged in the procedure.

Let's look at a small program using the objects in the sample screens above: a girl called suzie and a cat called kittycat.

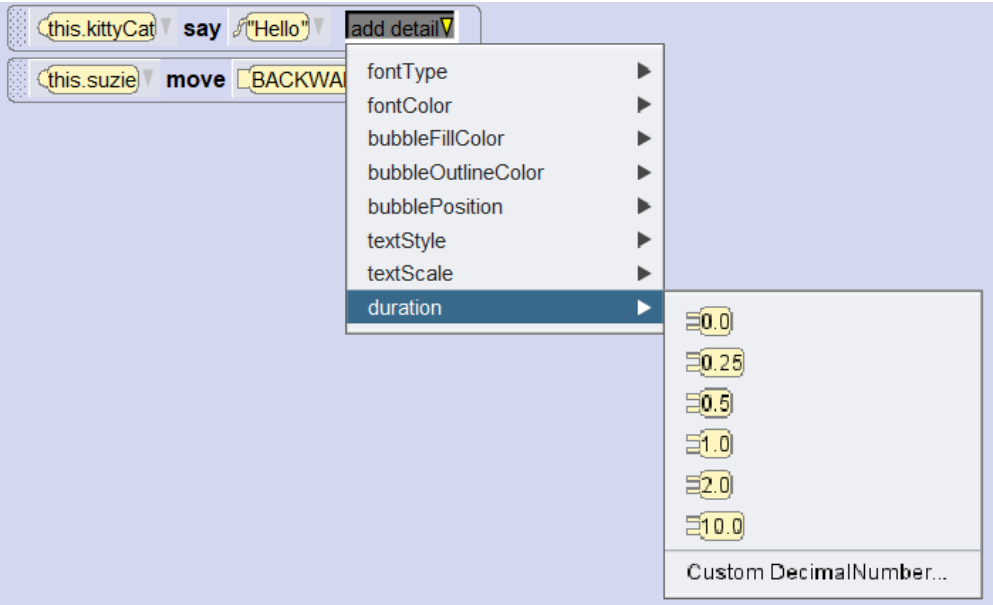
First we need to plan our action in detail, step by step. You need to think like a film director!

- Kittycat says 'Hello'
- Suzie moves backwards in fright
- Suzie says, 'A talking cat??'
- KittyCat says, 'Yeah, I can also fly. Wanna come fly with me?'
- Suzie turns to face KittyCat
- KittyCat turns to face Suzie
- Suzie says 'Yeah!'
- KittyCat jumps onto Suzie's shoulder.
- They fly away.

Now we use the code editor to program these steps.

We select kittyCat in the object finder. Drag the 'say' procedure into the editor. We'll be given the choice of clicking 'hello' or 'Custom text string'. Since we want 'Hello' with a capital 'H', we choose Custom Text String and type 'Hello' into the pop-up box.

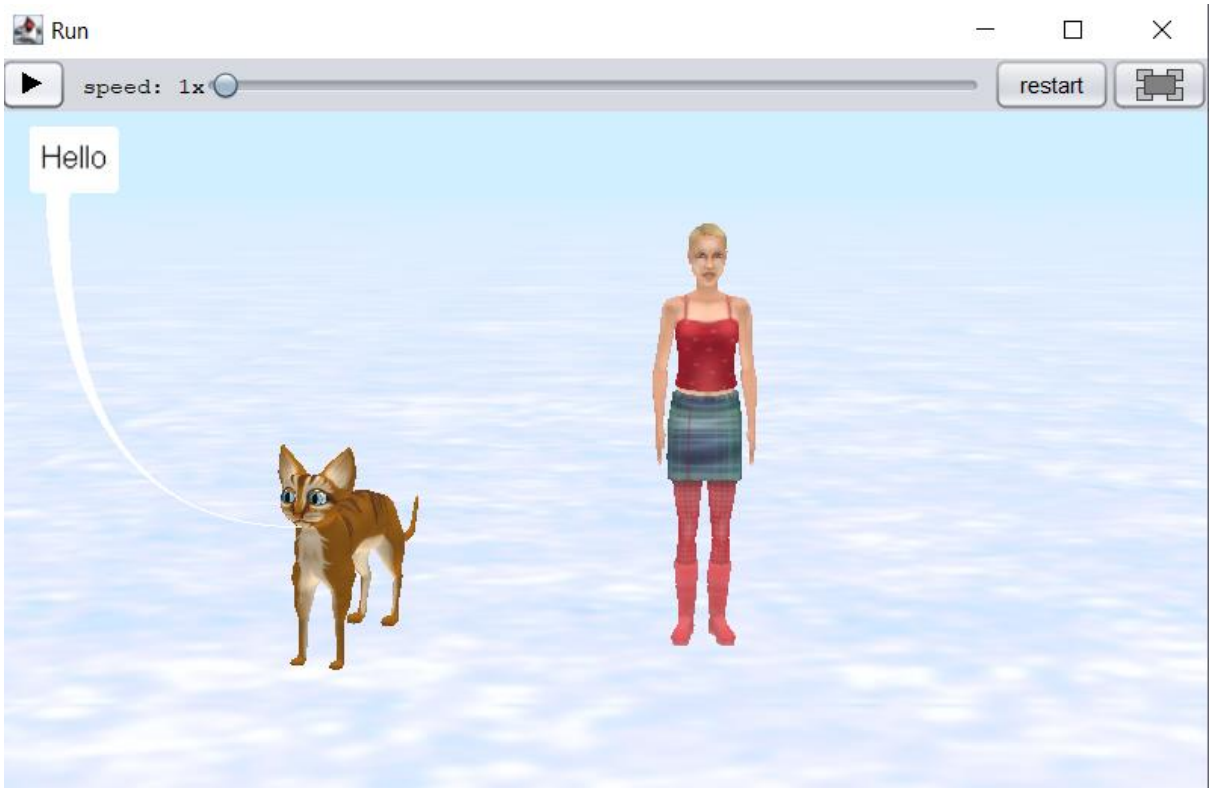
We'll adjust the duration of the 'say' procedure, to give the person watching the animation time to read it. We do this by clicking the 'add detail' box on the instruction and choosing 'duration'. A duration of 1.5 works nicely, but this isn't one of the choices, so we click 'Custom decimal number' and type 1.5 into the pop-up box.



Now we select suzie, and drag in the 'Move' procedure. We'll be given a choice of directions, then given a choice of how far to move her. We'll choose Backward by 1. Since we want this to happen quickly, we'll change the duration to 0.

We continue like this until all the steps have been programmed. kittyCat jumping onto Suzie's shoulder will involve several steps, including moving above her, orienting to face the same way etc. The final program might look like this.

Once the program is finished, you can click the 'Run' button to see what it does. In fact, it's probably a good idea to run it after entering every few steps of the program, to make sure it's really doing what you want it to. Clicking 'Run' runs the program in a pop-up box like this:



When you close the pop-up, you'll be returned to the program editor.

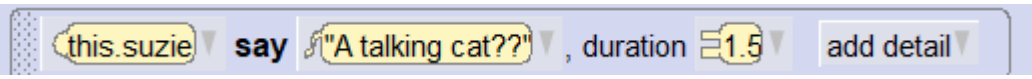
3.3 Making changes

When you run the program, you'll usually see things that aren't working quite right, and you'll want to change them. Creating a program is a process of setting the scene, writing the code, then repeatedly testing and tweaking until the program does exactly what you want. Nobody gets it right first time.

For example, after running this program, we might decide that some of the movements are too slow, and that the camera is too far away after Suzie moves backwards. Maybe it would also be nice to have a moment of semi-darkness while the magic happens just before the fly away.

In the code editor, you can change lines, delete lines, insert new lines, duplicate lines or change the order.

Changing a line: There are several things you can change in a line of code. If we look at this line:



There are arrows next to the object name (suzie), the text she will speak, the duration, and 'add detail'. This means that by clicking the arrow, you can change any of these things.

Inserting a line: Inserting is simple; just drag the new instruction in between the lines where you want it to go. To move the camera forward after Suzie has stepped back, select the camera object and drag in the 'move' procedure.

Moving a line: Position on the grey area to the left of the instruction you want to move, and drag the instruction to the new place.

Deleting a line: Right-click the grey area just to the left of the line, then choose delete.

Duplicating a line: To duplicate a line, right-click it, and choose Copy to Clipboard. The clipboard at the top right will turn white to show there is something in it. Now click on the clipboard and drag it to the place where you want the duplicated line to go.

If we want to add a moment of semi-darkness before the flight, we would:

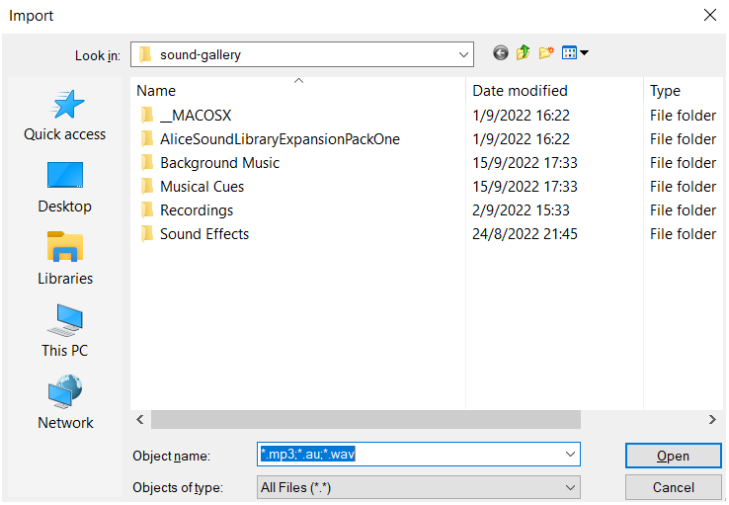
- Select 'this' from the object finder
- Drag in the setFogDensity to just after kittycat jumps onto suzie's shoulder, and set it quite dark – maybe about 7.
- Drag in the setAtmosphereColor to darken the sky; you can experiment with different colours.
- We then want to change the sky back again. Right-click the setAtmosphereColor command and copy it to the clipboard.
- Drag it back from the clipboard, then click the arrow next to the colour, and change the colour. We will have to choose 'Custom Color', and pick one closest to the original colour of the sky.
- Now copy the setFogDensity to the clipboard, drag it back, and change the copied instruction to change the fog density back to zero.

You could also experiment with the setAboveLight and setBelowLight to get more effects.

We now have a working animated story. It could be improved by making the action more smooth, and by changing the position of different body parts – for example, when kittyCat is on suzie's shoulders, he would look more realistic if we changed the position of his head and legs. Since this course is just a quick look at Alice, we don't have time to learn how to do all these things. They will be covered in the full Alice course.

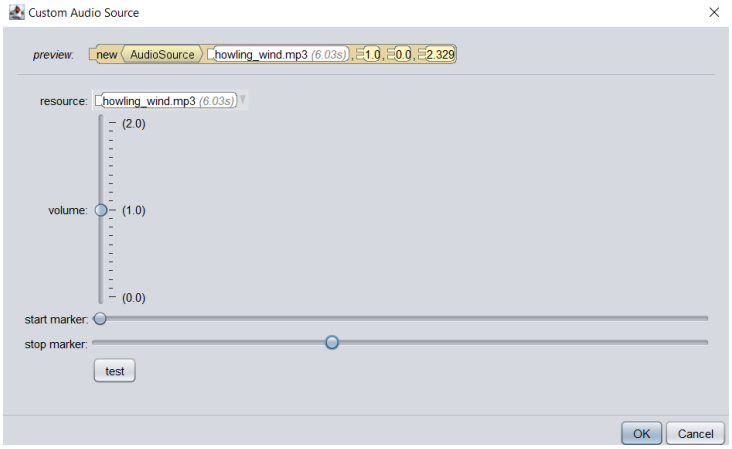
3.4 Adding Sound

The last thing we'll look at on this course is how to add some sound effects to your program. To do that, we use the playAudio procedure. Most of the objects have a playAudio procedure; it doesn't really matter which object you use. We'll drag this into the code editor and put it just before the setFogDensity command, and choose 'Import audio'. It then pops up a File Chooser box:



Alice provides standard sound effects in the 'Sound effects' folder. You can choose from these. If you downloaded the sound expansion pack, this gives you a choice of more sounds. You can also use other sound files you have on your computer, or record your own using recording software. In the full course, we'll learn how to record our own audio files with Audacity free software to create voice-overs. For now, we'll choose 'Sound effects', then 'Ambient', then 'Howling wind'.

If, when you've run the program, you want to change the duration of the sound, you can do this by clicking the arrow next to the audio source in the playAudio command. Choose Custom Audio Source. This will show this pop-up:



Here, you can change the start or stop time or the volume.

Please do Exercise 2 in the Practicals handout.

4 Preview of Next Course

From this course, you should have a good idea of what Alice programming is all about, and what you can do with it. If you enjoyed the course and want to go further, you may like to come on the full course. We will be learning, among other things:

- More about scene setting in 3-D
- Moving sub-parts of an object, eg arms, legs and head.
- Setting more than one scene in your world
- Using text, billboards and shapes

- Using repeated actions, conditional actions and do-together actions
- Writing procedures, and importing procedures from other programs
- Making programs interactive
- Detecting collisions
- Creating sound files with Audacity

Thanks for joining and hope to see you on the next course.