

# Princess & Dragon – Version 2

## Part 3: Billboards, Events, Sounds, 3D text and Properties



By Michael Hoyle  
under the direction of Professor Susan Rodger  
Duke University  
July 2012

# Overview

In this last part, we will show how to enhance your worlds. The main topics covered are **billboards**, **3D text**, **events**, **sounds**, and **properties**.

- **Billboards** are a way for you to convert 2D images into objects.
- **Events** add interaction into your world and tell Alice when to use your methods.
- **Sounds** can be imported and played in your Alice world.
- **3D Text** is an Alice object which allows you to add custom 3D messages to your world.
- All objects have **properties**, such as color, which can be changed.

# Making a billboard

Billboards provide a way for you to get an external image on your computer into your Alice world and turn it into an object. Billboards are different than other Alice objects because they are very thin. They are a flat image that can be moved and rotated in your 3D world, much like a real-life billboard.



We want the image on the left to be the background of our world. It should have come with this tutorial as 'castle.jpg'

# Making a billboard


To add a billboard to the world, click File > Make Billboard...

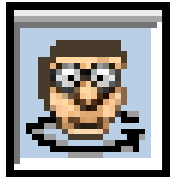
**Important: Do not click Import! That option is not for billboards**

A box will pop up asking you to choose the image you would like to import as a billboard. Find and choose 'castle.jpg' and click Import. It should appear as a small image in the center of your world, like below.

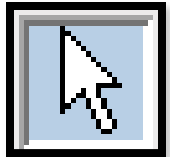


# Positioning the billboard

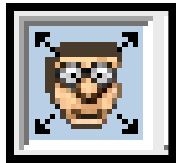
Because we want this castle as the background of our animation, we want it to be much bigger and further back. Luckily, like any other object, we can use our positioning tools to get it exactly where we want it. Click  to access the positioning tools.



Use the Turn tool to make it face the camera.



Use the Move Freely tool to push it farther in the background, at least behind the dragon.



Use the Resize tool to make it much bigger so it covers the entire background of the screen.

You may also need to use other tools to get the billboard exactly where you want it. Use whatever you need, you know how all of the tools work!

*Make sure the billboard is pushed far enough back in your world to give your objects room to move around in front of it.*

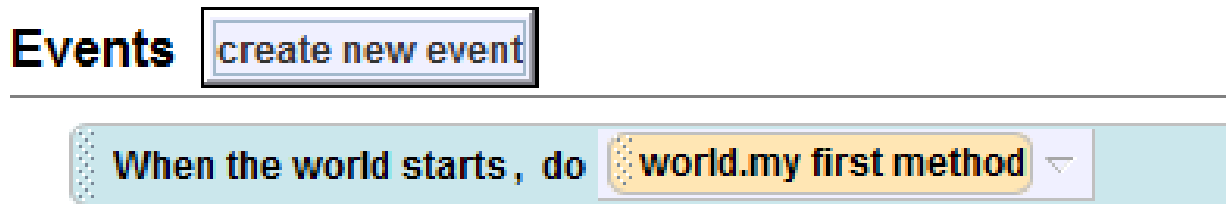
# Positioning the billboard

Remember that our animation has two different camera views, and we want the background to be in both. Change camera views by right-clicking the camera and selecting methods > set point of view to > camera views > knight view to check if the knight view is covered. If not, make the billboard bigger! **Make sure you set your camera back to 'original view' using the same method before you move on.** Once you're done, click 'Done'. Your scene should look like this:



# Events

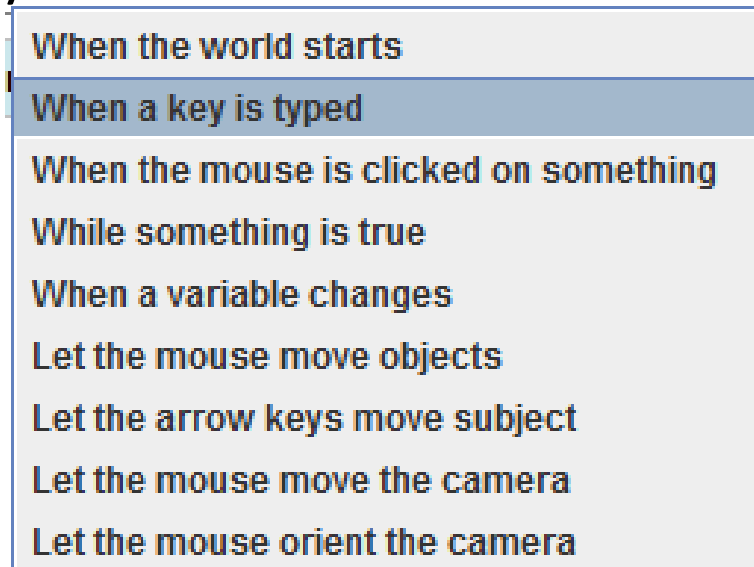
It's time for us to add another event to our world. Remember that **Events tell Alice when to use our methods**. Find the events panel in the upper-right part of Alice. There is already one event up there, remember it from part 1?



Without this event, our animation would do nothing! This tells Alice to run all the code in world.my first method when you press Play. That's why when we've written new methods, we've needed to add them to world.my first method, or else Alice never knows when to use them.

# Events

That event tells Alice to run a method when the world starts, but other events tell Alice to run methods when other things happen. Click 'create new event' to get a listing of all the events you can create.



You'll see a new event pop up. We want to change it to mean "When space is typed, have the dragon flap his wings"

**Events**

create new event

When the world starts, do world.my first method

When any key is typed, do Nothing

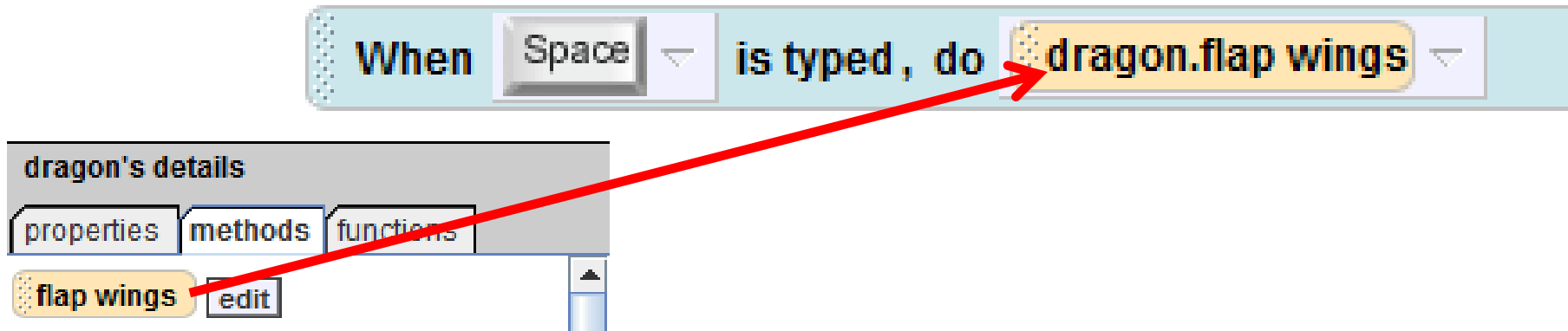


# Events - when a key is typed

Click the arrow beside 'any key' to change it to 'Space'. Now we need to tell it which method to do run when space is typed.

*Remember, events always tell Alice **when** to run a method*

Click the dragon and go to its methods tab. There you will find the 'flap wings' method we created in part two. Click that and drag it over to your event where it currently says 'Nothing'. Now your event should look like this:



# Press Play!

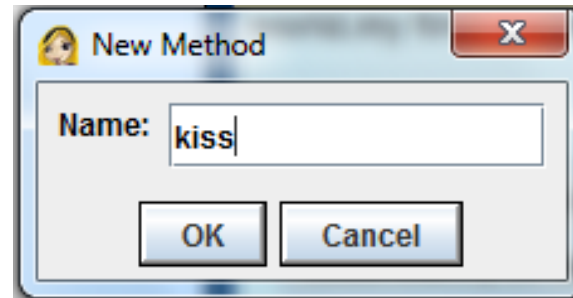
Test out your world. Try pressing space at any time, and your dragon should flap his wings.

Remember that you use the same method to tell him to flap his wings in the code, so sometimes he may flap his wings without you pressing space.

*Avoid typing the space key quickly or too many times in a row. Alice will remember how many times you typed the key, and the dragon will keep flapping his wings long after you stopped typing space.*

# The Final Scene – a Kiss

It's finally time for us to write the final scene of our animation, where our knight picks up our princess, kisses her, and then carries her off to live happily ever after.



We will write this in a new world-level method. Click **world** and go to the methods tab. Click **create new method** and call it “kiss”

# The Final Scene – a Kiss

You should be good at writing methods by now! Try to write the code for this script in your new world.kiss method:

1. Knight turn to face cinderella
  2. Cinderella turn to face knight
  3. Cinderella say “My sweet prince! I am saved!”
- } *Do together*

If you get stuck, you can find the correct code on the next slide.

# The Final Scene – a Kiss

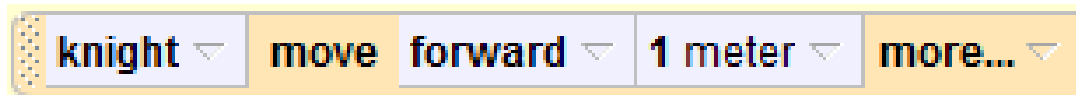
The image shows a Scratch code editor window with two tabs: 'world.my first method' and 'world.kiss'. The 'world.kiss' tab is active. Below the tabs, the text 'world.kiss No parameters' is displayed. Underneath, it says 'No variables'. The main code area contains a 'Do together' block with three sub-blocks: 1. 'knight' followed by 'turn to face' and 'cinderella'. 2. 'cinderella' followed by 'turn to face' and 'knight'. 3. 'cinderella' followed by 'say' and 'My sweet prince! I am saved!'.

Here's the right code.  
Notice how the  
cinderella say method is  
outside the Do together.

Our next step is to have our knight move to our princess. But if we use the 'move to' method, he's going to move right on top of her! *Remember that 'move to' always lines up the centers of the objects so they are in the same place.* We need him to move forward the distance in front of cinderella... but how?

# The Final Scene – functions to the rescue

How far in front of cinderella our knight is – that's a piece of information about our world. We can get it by using **functions**. We need our knight to **move forward**, but we don't know how far, so we let a **function** figure out this distance for us. Go to knight's methods tab and drag in a **move** method. Choose forward > 1 meter. Remember we don't actually want him to go one meter, we just choose that for now until we replace it with a function that computes the correct distance..



# The Final Scene – functions to the rescue

knights details

properties methods **functions**

create new functions

[-] proximity

- knights is within threshold of
- knights is at least threshold a
- knights distance to
- knights distance to the left of
- knights distance to the right of
- knights distance above
- knights distance below
- knights distance in front of**
- knights distance behind

Now go to knight's functions tab. Find **distance in front of**. Drag that over '1 meter' and select cinderella > the entire cinderella. Now no matter how far away he is from her, the function will figure out how far he should move!

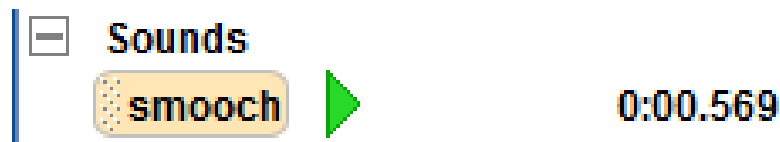
```
knight move forward knight distance in front of cinderella more... more...
```

# Adding Sound

Now that the knight is standing in front of the princess, they should kiss. For this, we're going to have our world play a smooch sound. There should be a file called "smooch.wav" included with this tutorial. Let's import it into our world so we can use it.

Click "File" and select "Import...". A dialog box will pop up asking you what you would like to import; find and select the smooch.wav file.

Seemingly nothing will happen, but the sound should be in your world and ready to use. Click **world** in the object tree and go to its properties tab. Near the bottom you'll find a section called "Sounds". Press the + to expand it and you should see "smooch".





# Adding Sound

Once you've imported a sound into your world, playing it in your animation is as simple as dragging and dropping into your code. Drag **smooch** into **world.kiss**.

The screenshot shows a Scratch code editor with a 'Do together' block containing the following actions:

- knight turn to face cinderella more...
- cinderella turn to face knight more...
- cinderella say My sweet prince! I am saved! more...
- knight move forward knight distance in front of cinderella more... more...
- world play sound world.smooch (0:00.569) more...**

*Your code for world.kiss should look like this at this point.*

# Finishing world.kiss

Now we just need to finish up our world.kiss method. To finish off, we'll have the knight pick up cinderella, spin her around, and take her away to live happily ever after.

To simulate picking cinderella up, we'll just have her **move up 0.5 meters** with a **duration = .25 seconds**. The code for this is below.

The image shows a Scratch code editor with several blocks in a script area. The blocks are as follows:

- A purple "Do together" block containing:
  - A yellow block: knight turn to face cinderella more...
  - A yellow block: cinderella turn to face knight more...
- A yellow block: cinderella say My sweet prince! I am saved! more...
- A yellow block: knight move forward knight distance in front of cinderella more... more...
- A yellow block: world play sound world.smooch (0:00.569) more...
- A yellow block, highlighted with a red border: cinderella move up 0.5 meters duration = 0.25 seconds more...

## Introducing properties – the vehicle property

Now that the knight has picked cinderella up, he is holding her. That means that wherever the knight moves, cinderella should stay in his hands and move with him.

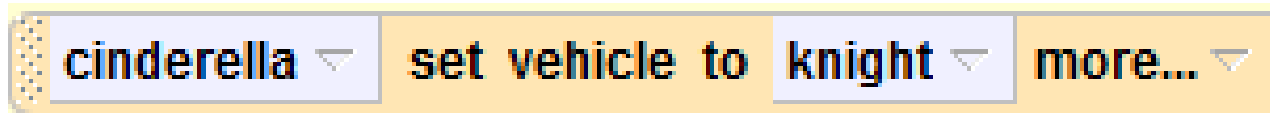
Alice gives us a simple way to do this with the **vehicle** property. Setting an object's **vehicle** is like **gluing it to another object**. So if cinderella's vehicle is the knight, she is glued to him and will move wherever he moves.

To find the vehicle property, go to cinderella's properties tab. Find vehicle, and it should say vehicle = world. That's the default, and means that she's not glued to anything.

*Note: the “gluing” only works one way. If cinderella's vehicle is the knight and she moves, the knight will not move with her.*

# Introducing properties – the vehicle property

Click **vehicle** and drag it into your code. When asked for a value select knight > the entire knight.



Now wherever the knight moves, cinderella will move with him, as if he were holding her!

*Note: To “unglue” cinderella from the knight (i.e. if we wanted the knight to let go of cinderella), just set cinderella’s vehicle back to **world**.*

# Finishing world.kiss

Almost done! Here's the script for the last part of world.kiss, try to do it on your own:

1. Knight turn right 1 revolution
2. Knight say "Off to my castle!"
3. Knight move forward 10 meters

*Remember that wherever the knight moves cinderella will too now. Even though we are calling methods on just the knight, it will look like he is twirling her around, talking to her, and then carrying her off.*

# Final code for world.kiss

[-] Do together

knight ▾ turn to face cinderella ▾ more... ▾

cinderella ▾ turn to face knight ▾ more... ▾

cinderella ▾ say My sweet prince! I am saved! ▾ more... ▾

knight ▾ move forward ▾ knight ▾ distance in front of cinderella ▾ more... ▾ ▾ more... ▾

world ▾ play sound world.smooch (0:00.569) ▾ more... ▾

cinderella ▾ move up ▾ 0.5 meters ▾ *duration* = 0.25 seconds ▾ more... ▾

cinderella ▾ set vehicle to knight ▾ more... ▾

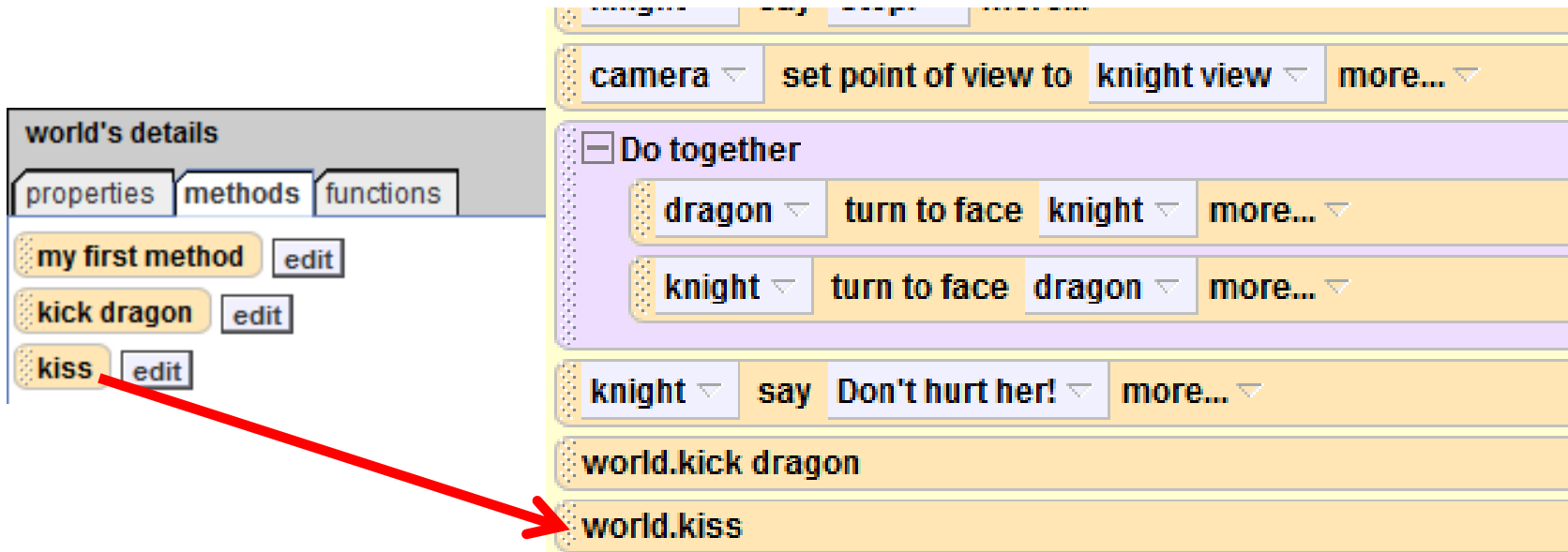
knight ▾ turn right ▾ 1 revolution ▾ more... ▾

knight ▾ say Off to my castle! ▾ more... ▾

knight ▾ move forward ▾ 10 meters ▾ more... ▾

# Back to world.my first method

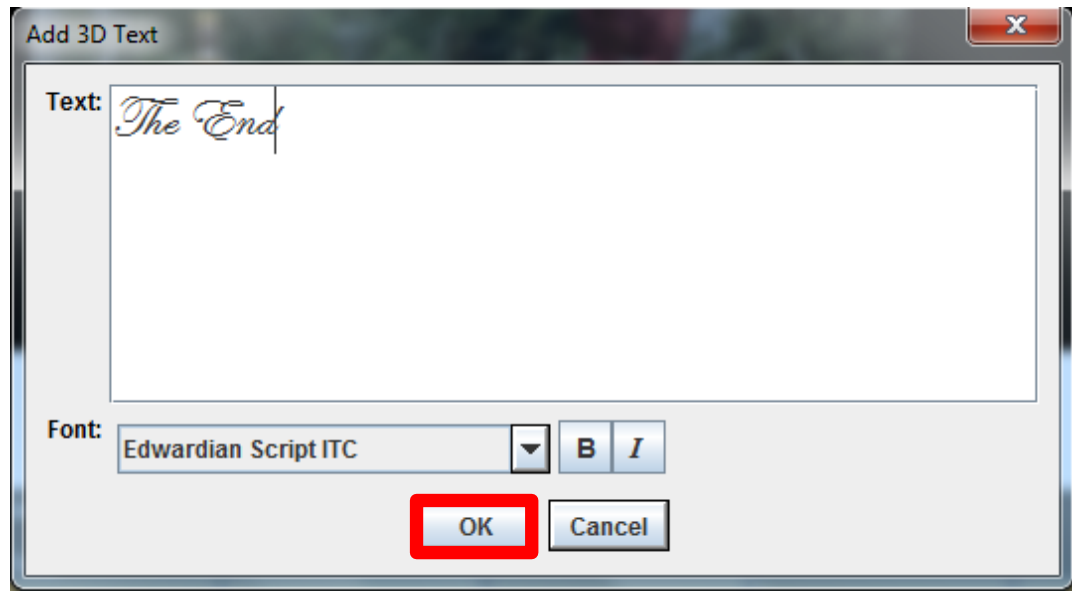
Always remember that just because you've written a method doesn't mean Alice knows when to use it. Return to your world.my first method tab to drag a **kiss** method below the current code.



# 3D Text

Our animation is essentially complete! Play your world and see if everything runs smoothly.

As a finishing touch, we will add some 3D text to say “The End” once the animation is complete. To add 3D text, click **Add Objects** and scroll all the way to the right to find “Create 3D Text”. Click it, and a box will pop up asking you which text and font you would like to use. Type “The End” for your text, and choose a font you think looks nice. **Edwardian Script ITC** is good if you can’t choose.





# 3D Text

Once you click Okay, The 3D text should appear in your world. Use your positioning tools to place it front and center, like below.

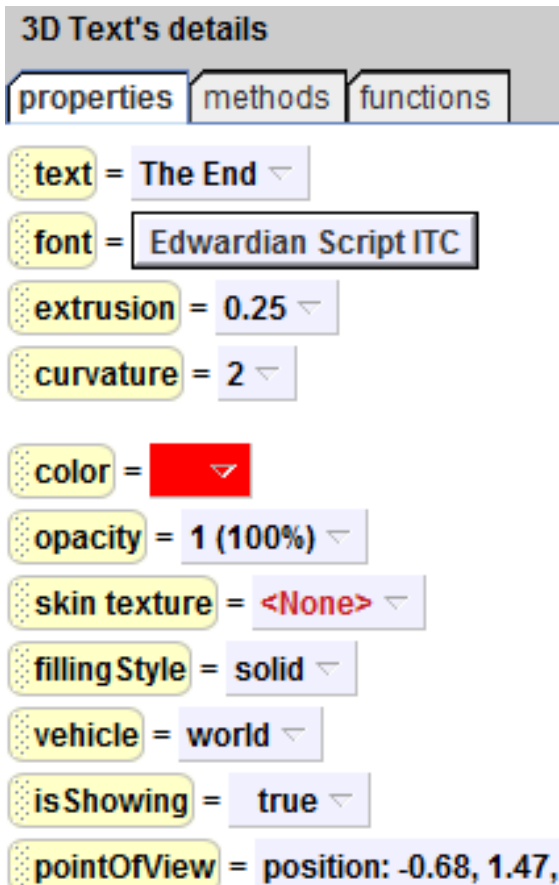


When the text is properly positioned, click Done.

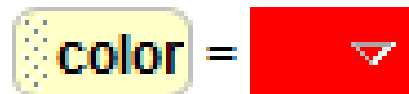


# 3D Text - Properties

Click **3D Text** in the object tree and go to its properties tab. There you will find the same properties as any other object with a few additional ones, such as text and font.



Find the **color** property and click the white box with an arrow. You should see a list of colors you can change it to. Select **red**, which will stand out more than white.



# 3D Text - Properties

3D Text's details


properties | methods | functions

text = The End ▾

font = Edwardian Script ITC

extrusion = 0.25 ▾

curvature = 2 ▾

color =  ▾

opacity = 1 (100%) ▾

skin texture = <None> ▾

filling Style = solid ▾

vehicle = world ▾

isShowing = false ▾

pointOfView = position: -0.68, 1.47,

Now find the **isShowing** property. This property determines whether or not an object is visible. Click the arrow next to “true” to change its value to “false”. The text should disappear in your world preview. This is good, we don’t want it to say “The End” for the whole animation!

isShowing = false ▾

## 3D text

To make your 3D text appear at the end of the animation, we just need to set **isShowing** to **true**. Click the **isShowing** property and drag it below the current code in world.my first method. When a popup asks you what value you would like to set it as, select **true**.

This means that the 3D text will remain invisible for the entire animation until the end, then it will appear.

You can find what the code looks like for this step, as well as the code for all of world.my first method on the next two slides.

# World.my first method – final code

The image shows a sequence of Scratch code blocks for a character named 'dragon'. The blocks are as follows:

- Block 1:** 'dragon' (object), 'turn to face' (action), 'cinderella' (target), 'more...' (dropdown).
- Block 2:** 'cinderella.cry for help' (action).
- Block 3:** A purple 'Do together' loop block containing:
  - Block 3.1:** 'dragon' (object), 'move' (action), 'up' (direction), '1 meter' (distance), 'duration = 2 seconds' (duration), 'more...' (dropdown).
  - Block 3.2:** 'dragon' (object), 'move' (action), 'forward' (direction), '2 meters' (distance), 'duration = 2 seconds' (duration), 'more...' (dropdown).
  - Block 3.3:** A light blue 'Loop' block with '2 times' (count), 'times' (type), and a sub-block:
    - Block 3.3.1:** 'dragon.flap wings' (action).
- Block 4:** 'dragon' (object), 'turn' (action), 'right' (direction), '0.25 revolutions' (amount), 'more...' (dropdown).
- Block 5:** 'dragon' (object), 'turn' (action), 'left' (direction), '1 revolution' (amount), 'asSeenBy = cinderella' (target), 'more...' (dropdown).
- Block 6:** 'cinderella.cry for help' (action).

*Continued on next slide*

# World.my first method – final code continued

knight ▾ say Stop! ▾ more... ▾

camera ▾ set point of view to knight view ▾ more... ▾

[-] Do together

dragon ▾ turn to face knight ▾ more... ▾

knight ▾ turn to face dragon ▾ more... ▾

knight ▾ say Don't hurt her! ▾ more... ▾

world.kick dragon

world.kiss

3D Text ▾ set isShowing to true ▾ more... ▾

# Congratulations!



Play your world and see what you've created! You now know all the basics of Alice and are ready to move forward and create your own unique worlds!

# Additional Challenges

Want to expand on your world? Here are a few challenges you can try:

- Make the knight sit on the horse more realistically. You'll have to rotate his legs (hint: this is described in the longer Princess Dragon tutorial), but remember that anything you do to his legs in setup you'll have to undo in your code before he kicks the dragon.
- Instead of having the knight and princess walk off screen at the end, have them move towards the castle in the background and fade into the distance (hint: you'll have to use the opacity property)
- Make the dragon breathe fire (hint: there are fire objects in the Special Effects folder)
- Make the fight scene between the knight and dragon longer and cooler – get creative!



# Learning More

This is only the beginning of the tutorials offered in Duke University's Alice Materials Repository. In these tutorials you can learn more about...

- Events, including many types not covered in this tutorial
- How to make decisions (with If/Else statements)
- How to ask the user for answers to questions
- Adding lighting to your world
- Creating good scene changes by fading in/out.
- Using loops and lists

And much more! Check out all of our tutorials at:

<http://www.cs.duke.edu/csed/alice09/tutorials.php>

And even more example worlds and videos here:

<http://www.cs.duke.edu/csed/alice09/index.php>