3D animation

AnyLogic supports both 2D and 3D space in simulation models, and enables you to create high-quality interactive 3D animations in addition to more technical-looking 2D animations. You can define a 3D scene, use the standard shapes provided in the 3D palette, imported 3D graphics, or include 3D objects composed of primitive shapes you create yourself. You can associate the 3D objects with entities, pedestrians, rail cars, and vehicles. Agents can live and move in 3D space. You can view 3D animation in one or multiple 3D windows simultaneously with 2D animation. 3D animation works everywhere: when running the model from within the AnyLogic development environment, exported as a Java application, or published on the web as a Java applet.

AnyLogic graphical editor is two-dimensional, and the natural scenario of building a 3D animation starts with a 2D (X,Y) plane, upon which you draw the "XY projection" of the scene, and then "grow" the picture into the third, Z-dimension (see the Figure). The models originally designed as 2D can be converted into 3D easily by defining the Z-properties of the 2D shapes.

Creating a 3D animation

There are two palettes for 3D animation: 3D and 3D Objects (see the Figure). The first one contains primitive shapes and 3D-specific objects, such as 3D window or Camera.
The second palette contains frequently used 3D graphics such as a person, car, forklift truck, house, etc., that can be associated with static or dynamic objects in the model.

3D-related palettes

Example: A very simple model with 3D animation

We will create a very simple model with 3D animation. In this model, people will exit a house and walk to a store located nearby.

Follow these steps:

1. Create a new model.
2. Put together a simple flowchart as shown in the Figure. Select **Maximum capacity** for the delay object, and leave all other parameters as is.
3. Open the 3D palette and use it to draw a background rectangle and a polyline as shown in the Figure. The shapes dropped from that palette differ from the same shapes from the **Presentation** palette only in that they have the property **Show in 3D scene** selected by default.
4. On the **Advanced** property page, set **Z** and **Z-height** of the shapes, as shown.
5. Open the **3D Objects** palette and drag the **House** and the **Store** objects on the background rectangle.
6. From the same palette, drop the **Office Worker** anywhere on the canvas. It looks like a bug, but don’t worry; this is the top view of the office worker.
7. Set the **Entity animation shape** of the **source** object to **officeWorker** and set the **Animation guide shape** of the **delay** object to **polyline**.
8. Drag the **3D window** from the 3D palette and place it below the flowchart.
9. Run the model.
The "source code" of a simple model with 3D animation

At runtime, you are able to view 2D and 3D animations simultaneously. In the 3D window you can drag the scene, use the mouse wheel to zoom, or use Alt+drag to rotate the view.

"A very simple 3D model" at runtime
Primitive 3D shapes

You may have noticed that the primitive shapes in the 3D palette are the same as in the Presentation palette. Indeed, they are in fact the same objects, with the only difference being that their Show in 3D scene property is selected by default. Therefore, by default, they will appear in the 3D scene.

In 3D, the rectangle shape will become parallelepiped (a solid whose faces are all parallelograms) and the oval will become a cylinder (see the Figure). The Z-coordinate of their bottoms, along with their Z-heights, are defined on the Advanced properties page. Do not confuse the Height of the rectangle and its Z-Height: the former actually becomes “Y-Height” in 3D.

3D Rectangle and 3D Oval. Z and Z-Height properties

Lines, polylines, and arcs have additional 3D properties. Lines and arcs have \( dZ \) – the difference in the Z coordinate of their start and end points. For a polyline, you can specify the Z-coordinate of each point in the Points property page. The 3D versions of these shapes are shown in the Figure.
3D Line, 3D Polyline, and 3D Arc. dZ and Point Z properties

If **Line color** is set for a 3D shape, it becomes the color of its "walls," both outside and inside. **Fill color** becomes the color of its "bodies" (see the Figure).

Please note that if any two points of a polyline or start and end point of an arc have different Z coordinates, these shapes **cannot be filled in 3D**. They can only be filled if they have "flat surface."
Line and fill colors of 3D shapes

Images and text can also appear in a 3D scene. Both have no Z-height (they are infinitely thin) and can be viewed both from the top and from underneath.

If two or more shape surfaces happen to be in the same plane and they intersect, the 3D renderer will not know how to display them. Undesired visual effects may then occur. To avoid this, you should slightly shift one of the surfaces by making the corresponding coordinates a bit different.

Consider the Figure. To make the text appear nicely on top of the map, its Z coordinate was set to 1. Sometimes, one unit of difference is too much; in that case, you can set it to a fraction, say, 0.2 (this can be done in the dynamic properties of the shape).
The 2D order of shapes (the order that is controlled by the Send to Back/Bring to Front commands) does not affect the 3D scene at all. The shape that appears underneath another shape in 2D can be above it in 3D!

The 3D-related properties of shapes can be dynamic and can be linked to model variables and expressions, just like all other properties (see the Dynamic property page of a 3D shape). The Rotation of a shape is called Rotation Z, if the shape is marked as 3D. This is because the 2D rotation is the rotation in the (X,Y) plane around the Z axis.

You will find no fields for rotation in the (Y,Z) plane ("Rotation X") and rotation in the (X,Z) plane ("Rotation Y"). These cannot be controlled for individual shapes, but can be controlled at the 3D group level.

3D groups and rotation

3D shapes can be grouped in the normal way. The group’s Show in 3D scene property "owns" the same property of all the group members. If a shape being grouped is in 3D, the group is also marked as 3D, and so are all other shapes in the group. Therefore, either all shapes in a group are shown in a 3D scene, or none of them are.

3D groups have dynamic Rotation X and Rotation Y properties, which allows you to rotate 3D shapes in the (Y,Z) and (X,Z) planes – this is the only place where you can do it.

Example: Rotation in 3D – a sign on two posts

We will draw a sign with two posts. At design time, the sign will lie on the ground, i.e. in the (X,Y) plane. However, at runtime, it will be raised vertically by using the 3D group rotation properties.

Follow these steps:

1. Create a new model.
2. Draw the picture (as shown in the Figure) using the text, rectangle, and line objects from the 3D palette. Set the Z-properties of the shapes as shown.
3. Select all shapes and group them. A new group is created, with the center approximately at the center of the picture.
4. Click the group and choose Select group contents from the context menu.
5. Move the group members up so that the group center is at the bottom of the posts (see the Figure on the right).
6. Select the group and open its Dynamic properties page. Set Rotation X to -PI/2.
7. Drop a **3D window** anywhere on the canvas. In its **Scene** properties page, set **Grid color** to **white**.

8. Run the model. The sign is displayed as being raised up, but the text appears distorted because its surface intersects with the surface of the sign (text's $Z = 2$ and it has no Z-height, and the sign's $Z + Z$-Height = 2 as well).

9. To fix this, open the **Dynamic** page of the text properties and set **Z** to **2.1**. This will position the text slightly above the sign surface.

10. Run the model again.

### Example: Bridge crane 3D

We will build a 3D model of a bridge crane. The bridge crane has three degrees of freedom: it moves along the rails (in the X dimension), its hoist moves along the bridge (the Y dimension), and the load moves up and down (the Z dimension). The crane will be controlled by three sliders. By moving a slider, you can define the target coordinate for X, Y, or Z, and the crane will start moving. We will use a number of 3D shapes along with a 3D group with dynamic properties.

#### Follow these steps:

1. Create a new model. Open the **Window** page of the **Simulation** experiment properties, and set the dimensions of the window to 900 x 700.

2. Open the editor of the **Main** object. Draw the crane "XY projection" with three 3D rectangles and two 3D circles, as shown in the Figure.

3. Group all these shapes. Move the group contents, and then the group, so that the group center is at the coordinate origin, as shown the Figure on the right.
The (XY) projection of the bridge crane

4. Drop the 3D Window from the 3D palette. Move its top left corner to (0,400) and resize the window to 900 x 300 pixels.

5. Run the model and view the (static at this point) bridge crane.

- Create the model of the crane movement in X dimension

6. Add a parameter speed with a default value of 50. For simplicity, we will use the same speed for movement in all three dimensions.

7. Add an event moveX, two variables, startX and targetX, and a function X(), as shown in the Figure.

The model of crane movement in X dimension

The function X() returns the current X coordinate of the crane. If the crane is not moving (meaning the event moveX is inactive), the current X equals the target X. Otherwise, we calculate the distance to cover, which is the product of speed and time.
to target, and subtract it from the target X. The slider sets the start X to the current X and starts a new movement by scheduling \texttt{moveX} at the time of arrival.

8. Select the group and define its dynamic X as \texttt{X()}, so that the X coordinate of the crane animation always equals the current X of the crane.

9. Run the model and move the slider. The crane moves along the X axis. If you move the slider while the crane is in motion, it cancels the previous command and moves to the new target location.

![Bridge Crane 3D](image)

Dynamic properties of the hoist

- **Add movement in the Y and Z dimensions**

10. Create two copies of the five objects \{ event \texttt{moveX}, variables \texttt{startX} and \texttt{targetX}, function \texttt{X()}, and \texttt{slider} \}.

11. Rename the objects in the first copy by replacing "X1" with "Y", and in the second copy by replacing "X2" with "Z", so that you have \texttt{moveY}, \texttt{moveZ}, etc.

12. Edit the code of events \texttt{moveY} and \texttt{moveZ}, functions \texttt{Y()} and \texttt{Z()}, and the two sliders, respectively, by replacing "X" with "Y" and "Z".

13. Set the range of the slider for Y to 0-200, and the range for Z 0-95.

14. Specify the dynamic properties of the circles that represent the hoist and the load, as shown in the Figure. Use the \texttt{Projects} tree to select small shapes.

15. Run the model and play with the crane.
Standard and imported 3D graphics

Primitive 3D shapes are primarily used to create simple 3D objects, or objects that dynamically change color, size, or have moving parts, like the bridge crane in the previous section. To create more realistic (and attractive) 3D animations, you can use sophisticated 3D graphics (also called 3D models), both standard included in AnyLogic, as well as your own.

Using standard 3D graphics

A collection of pre-drawn 3D graphics are available for your use in the 3D Objects palette (see the Figure).

To use a standard 3D graphics:

1. Drop the 3D object from the 3D Objects palette on the canvas.
2. Choose the scale and orientation (the Axis order property)

By default, the objects in the palette are put on the (XY) ground plane, and are therefore seen in the graphical editor from the top. However, you can change this (see the Figure). The default scale of objects is chosen to match the default library settings. For example: people, cars and trucks, trolleys, and security control devices are all in the same scale, which is consistent with the default scale of the Pedestrian Library. The scale of the rail cars and the containers matches the default scale of the Rail Library.
Orientation of 3D objects

Using external 3D graphics

AnyLogic uses X3D file format for 3D objects. X3D is the ISO standard XML-based file format for representing 3D computer graphics, and is the successor to the Virtual Reality Modeling Language (VRML) [Wikipedia]. A good collection of X3D resources can be found at [http://www.web3d.org/x3d/content/examples/X3dResources.html](http://www.web3d.org/x3d/content/examples/X3dResources.html).

While X3D does not have as many 3D models in public access as, say, Google Sketchup™, there are free and commercial conversion tools: Vivaty Studio™, Blender™, and Google Sketchup Pro™, for example.

To import Google Sketchup graphics into AnyLogic using Vivaty Studio:

1. Download Vivaty Studio from [http://www.web3d.org/realtime-3d/x3d-vrml/vivaty-studio-download](http://www.web3d.org/realtime-3d/x3d-vrml/vivaty-studio-download) and install it. This is free of charge.
2. Go to [http://sketchup.google.com/3dwarehouse/](http://sketchup.google.com/3dwarehouse/) and choose a 3D model to download. The model should be downloadable in either the Google Earth (.KML and .KMZ) or Collada (.DAE) formats; Vivaty Studio cannot read the .SKP file directly.
3. Save the Sketchup 3D model. If it is in Collada format, it is usually a ZIP archive, which you will then need to unzip.
4. Run Vivaty Studio and create a new project.
5. Choose File | Import Collada or Import Google Earth, depending on what you have downloaded. Import the model.
6. Choose **File | Save** to save the model (this is required before you can export it in X3D).

7. Choose **File | Export X3D or VRML**. In the **Save As** dialog select the **Uncompressed** option. Save the X3D file.

8. In AnyLogic, open the 3D palette and drop the **3D Object** on the canvas.

9. In the 3D object properties, choose the X3D file saved by the Vivaty Studio. Choose the appropriate scale and orientation.

Similarly, you can import the Sketchup graphics using the **Blender** tool.

**Hierarchical 3D animations. Embedded 3D presentation**

Just like with 2D animation, 3D animation in AnyLogic is hierarchical. The 3D presentation of an embedded active object can appear in the 3D scene of its container object. The embedded object presentation has the **Show in 3D Scene** and the **Z** properties.

**Embedded 3D presentation**
The point of origin of the embedded airplane presentation is placed into its (X,Y,Z). Therefore, if you ask the airplane to move to (-100, 0, 100), it will arrive at (100, 100, 150) on Main.

The airplane active object has its own 3D scene (in our case, it only contains the airplane picture). This can be viewed by adding a 3D window in the Airplane editor and navigating to the airplane object at runtime.

**3D Windows**

To view 3D animation in AnyLogic, you use 3D windows. The 3D Window object is located in the 3D palette. In the Scene properties page of the 3D window, you can set up the background color and (optionally) the grid color. These properties are shared by all 3D windows in the active object class. The grid is drawn on the (X,Y) plane (see the Figure) and has a step of 100 pixels.

**Navigation in the 3D scene at runtime**

If you do not associate the 3D window with a camera, in the beginning it will take the default view on the 3D scene, trying to display the entire scene from some perspective. You can change the viewpoint by using the mouse and the Alt key (see the Figure).
Navigation in the 3D scene

Keep in mind that, for performance optimization purposes, very distant objects are not displayed by the 3D rendering engine.

Sometimes you may wish to limit navigation. For example, you may not want the user to zoom in and out or to look below the ground level. You can limit navigation by choosing the appropriate Navigation type in the General page of the 3D window properties.

Multiple 3D views

While you have a single 3D scene per active object, you can have multiple views of the scene in multiple 3D windows.

Add a second 3D window to the simple model we created earlier

1. Open the model A very simple model with 3D animation.
2. Drag the second 3D window from the 3D palette and drop it on the right of the first one.
3. Run the model and try to navigate in the second window so you see the people entering the shop through the shop door (see the Figure).

Multiple views of the 3D scene

If the windows are not associated with cameras, in the beginning they all will display the scene from the same viewpoint. The viewpoint can then be changed individually for each window by using the mouse. To get different default viewpoints, you should link the widows with cameras. This is explained in the next section.

Cameras

Camera in the AnyLogic 3D toolset is a viewpoint that can be associated, both statically and dynamically, with a 3D window. There may be an arbitrary number of cameras in different active objects in the model. In particular, you can place a camera on a moving object and see things from its perspective. There may be several points of particular interest in your system such as known bottlenecks or sensitive equipment. You can assign a camera to each of them so as to quickly switch between these sensitive areas to assess the impacts of your simulation under different parameters or at different times.

Example: A very simple model with multiple 3D windows and cameras

We will use the first model we created in this chapter, and add the second 3D window and two cameras. The first one will be used as a default view for the first window, and the second one as the fixed unchangeable view for the second window.

The easiest way to tune the camera position is run the model, navigate a 3D window to the viewpoint you need, copy the viewpoint settings, and paste them into the camera properties in the editor.

Follow these steps:

1. Open the model A very simple model with 3D animation.
2. Reduce the width of the 3D window to leave space for another one.
3. Drag the second 3D window from the 3D palette and drop it on the right side of the first window.
4. Drag a Camera object from the 3D palette and drop anywhere.
5. Run the model.
6. Adjust the view in the first 3D window so you can see the full scene at a nice angle.
7. Right-click the 3D window and choose Copy camera location from the context menu (see the Figure).
8. Return to the model editor and select the camera. Press the Paste button in the camera properties (see the Figure). The camera will jump to a new location, and its rotation angles will also change.

Tuning the camera position

9. Select the first 3D window. In its properties, set the camera to the camera you have created. Do not select the Follow camera checkbox. Set Navigation type to Rotation only.
10. Run the model. The window now shows the scene from a different viewpoint. You are able to adjust it slightly, but you cannot zoom in or out.
11. Add another camera and use the same procedure (steps 5-9) to set the view in the second 3D window to the view from the shop door, focusing on the people entering the shop.
12. In the properties of the second 3D window, select Follow camera and set Navigation type to None.
13. Run the model. Test navigation in both windows.
Example: Camera on a moving object

There are two ways you can move a camera. First, you can define dynamically changing coordinates and rotation in the camera Dynamic property page. Secondly, you can put the camera on a moving object. In this example, we will have an airplane with a camera that will fly over the US territory. We will use the function setCamera() of the 3D window to switch between the static and moving cameras.

First, create a 2D model:
1. Create a new model.
2. Create a new active object class, Airplane. Select the Agent option in its properties.
3. Drag the Airliner object from the 3D objects palette and place its center at (0,0). Rotate the airliner object by 180 degrees so that its nose looks to the right, and set its scale to 50%.
4. Open the editor of Main. Drag the Airplane object from the Projects tree to the canvas. Now Main has an embedded object airplane. Its embedded animation is located at (0,0). Move the airplane animation to (200,50).
5. Open the Pictures palette and drag the USA Map object to Main. Move the map so that its top left corner is located approximately at (200,50). Send the map to the back so that it gets below the airplane animation, as shown in the Figure.
6. Open the editor of the Airplane class again and go to the Agent page of its properties. Write the following code in the On arrival section:
   moveTo( uniform( 450 ), uniform( 300 ) );
   This code sends the airplane to a new random location within the US map once it has arrived at the previous one.
7. Below, in the same property page, set Velocity to 50.
8. Copy the piece of code in On arrival and paste it in the Startup code field in the General property page. This way, we will cause the airplane to make the very first move.
9. Run the model and watch the airplane fly in 2D.
Now, add the 3rd dimension to the model:

10. Open the editor of Main, select the US map and select Show in 3D scene in its properties.

11. Select the animation of the embedded airplane, and set its Z to 30 on the Advanced property page. The airplane will fly at this altitude.

12. Drag the 3D window from the 3D palette and drop it below the map.

13. Run the model and see the airplane flying in 3D.

Put a camera on the airplane:

14. Open the editor of Airplane and drop in the Camera object from the 3D palette. Place the center of the camera slightly behind the tail of the aircraft and rotate it so that it looks toward the nose (see the Figure).
15. On the General page of the camera properties, set Rotation X to 20 degrees. On the Advanced page, set Z to 20. This raises the camera above the airplane and bends it a bit down so that we can see the airplane itself.

16. Open the editor of Main and add a Button from the Controls palette. Label the button “Airplane camera”. Type the following code in the button Action field:
   
   window3d.setCamera( airplane.camera, true );

   This code switches the 3D window to the airplane camera. The boolean argument true tells the window to follow the camera.

17. Run the model, watch the 3D animation, and then press the button. The 3D window now shows the view from the (virtual) point behind and above the moving airplane (see the Figure).

![View from the airplane camera](image)

View from the airplane camera

We will now add a static camera and provide the ability to switch between the airplane camera and the static camera.

- **Add a static camera:**

  18. Open the editor of Main and drop the Camera object from the 3D palette in the bottom left corner of the map.

  19. Add a second Button from the Controls palette and label it "Static camera”. Type the following code in the button Action field:

      window3d.setCamera( camera, false, 500 );
      window3d.setNavigationMode( NAVIGATION_FULL );

  20. Run the model and switch between the static and moving cameras.

The first line of code in the button action switches the 3D window to the static camera. The middle argument false tells the window not to follow the camera, and the last optional argument defines a smooth 500 millisecond transition between the views (you may add the same smooth transition to the code of the first button). The second line of code restores the ability to fully navigate within the 3D space.
Lights

To render the 3D scene, the 3D engine must know the sources and types of light. By default, the AnyLogic 3D engine assumes the presence of ambient light – the light that is everywhere (has no particular source), has no color, no direction and does not fade over distance. You can define your own lights in the 3D scene that will replace the default light. The lights can be of different types (such as point, spot, directional) and can have customized color and attenuation. They can also be static as well as moving.

In total, there can be up to 6 lights per 3D scene, including the lights that come with the embedded objects.

Light types and properties of a spot light

The Light object is located in the 3D palette. In the properties, you can choose the type of light. The available types are:

- **Ambient** – ubiquitous light that has no particular source and no direction, like the light on a cloudy day. The default light that is assumed in the scene without any light objects is ambient gray.
- **Directional** – light from a source that is so far away that the rays are parallel, like sunlight.
- **Point** – light from a point source that shines evenly in all directions.
- **Spot** – light from a point source that shines only within a cone of a given angle.

A light of any type has three color components:
• **Ambient** – the color of the ubiquitous component of the light, if any.
• **Diffuse** – the color of the direct light that falls on the object surfaces.
• **Specular** – the color of the light reflected from the object surfaces.

All light color components interact with the object colors. If a light component is white or gray (evenly includes the whole spectrum), the objects will show in their natural colors. If, for example, the light color is pure red and the object color is pure blue, the object will not be visible. The purple object in the red light will show dark red, because only the red component of its color will interact with the light. The Figure shows six boxes in the directional light with different color components.

**Six boxes in the light with different color components**

Point and spot lights have the **Attenuation** property. **Attenuation** is the effect of light diminishing over distance. Constant attenuation means the light intensity is totally unaffected by distance. The intensity of the light with linear attenuation is inversely proportional to the distance from the light source, and quadratic attenuation means a very sharp decline of light.
Example: Examples of Lights in 3D Scene

We will create a simple 3D scene with a road and a building with a gate. We will illuminate the road with streetlights and put an indoor spotlight inside the building. We will also create a car with headlights on it, and let it drive along the road and into the building.

Create a static 3D scene

1. Create a new model.
2. Use the 3D palette to create a scene, as shown in the Figure. Add some men in black suits (office workers from the 3D Objects palette) to the corner of the building.
3. Add a 3D window, set its Background color to black, and run the model. Remember how the scene looks like in the default light.

Add static lights

4. Add a Light object from the 3D palette and place it anywhere. Choose the Ambient light type, set the Ambient color component to darkGray, and reset two other components.
5. Run the model again. The scene is now much darker, as we have replaced the default light with very dark ambient light.
6. Add two more lights of the Point type and place them nearby the road (see the Figure). Set their Diffuse color component to darkGoldenrod, and reset two
other components. Set the Quadratic attenuation to 0.0002 and the Constant and Linear attenuation to 0. Set the lights’ Z to 50 on the Advanced property page.

7. Run the model and view the scene. You can see the road lit by the two streetlights. Quadratic attenuation creates the effect of light locality.

8. Add another light, this time of the Spot type, and place it in the bottom left corner of the building. Leave only the Diffuse color component set to slateGray and set the Cut off angle to 45 (this makes the light cone wider). In the Advanced property page set its Z to 50, and Angle X to 55 (this rotates the light code downward).

9. Run the model and see the cone of the spotlight inside the building.

Add a car driving with headlights

10. Add a new active object class to the model and call it Car. Make the Car an agent by selecting the corresponding checkbox.

11. Drag the Car object from the 3D objects palette and drop it near the coordinate origin.

12. Add a light of the Spot type, place it at the car windshield, and rotate it so that it looks forward (see the Figure). Set the Diffuse color to lightSteelBlue. On the Advanced properties page, set Z to 10 and Angle X to 5.
13. Open the editor of Main and drop in the Car object from the Projects tree. This creates an embedded object car, and its presentation appears on Main.

14. In the Startup code of Main, write:

```java
car.moveTo(670, 165, road);
```

here, road is the name of the road polyline.

15. Run the model.