

Blitz3D

Acceleration, Deceleration and Breaking  
Tutorial for Beginners

This tutorial is aimed at Blitz3D beginners and gives a quick and easy example of using acceleration, deceleration and even breaking on a model of a car wheel. A simple understanding of Blitz is recommended but most of the stuff featured in the tutorial will have been covered in the Blitz Beginner tutorials.

You will need a model of a wheel (basically a shallow cylinder) and a texture. Alternatively you can create a cylinder manually in the engine using the [CreateCylinder\(\)](#) command

## **Intro**

Start a new BB file and set up the engine with the following properties;

```
Graphics3D 800,600,0,2
SetBuffer BackBuffer()
AppTitle "Wheel Example"
```

This should be pretty straightforward if you've read the beginner tutorials.

Next we'll load the wheel mesh and the texture we'll be using. Ensure to set the directories to ones on your machine to prevent errors;

```
;LOAD WHEEL
global wheel=LoadMesh("wheel.3ds")      ;mesh
PositionEntity(wheel,0,0,0)              ;position the mesh
wheel_tex=LoadTexture("tire_texb.tga")  ;texture
EntityTexture(wheel,wheel_tex)           ;texture the wheel
```

We set the wheel entity to be 'global' so that we can reference the file from within a Function that runs outside of the main loop. Otherwise we would get a 'Memory Access Violation' (MAV) error when we try and run it. This is usually caused by media that Blitz is looking for but can't find. If you get one, check your media directories and file extensions for any errors.

Usually you would want to create variables for the wheels X,Y and Z coordinates so you can easily adjust the wheels' position later, but for this example placing it at 0,0,0 will suffice.

Next we'll create a camera, position it in the engine and change the background colour of the engine so we can see the wheel better;

```
;CAMERA
camera_piv=CreatePivot()
camera = CreateCamera(camera_piv)
PositionEntity(camera_piv,-20,0,20)
PointEntity(camera,wheel)
CameraClsColor(camera,100,99,139)
```

The camera pivot 'camera\_piv' is the parent of the camera defined by putting the name of the pivot in the brackets following the camera. By using a pivot you can offset the camera but rotate the pivot so the camera spins round an object, as seen in 3<sup>rd</sup> person games.

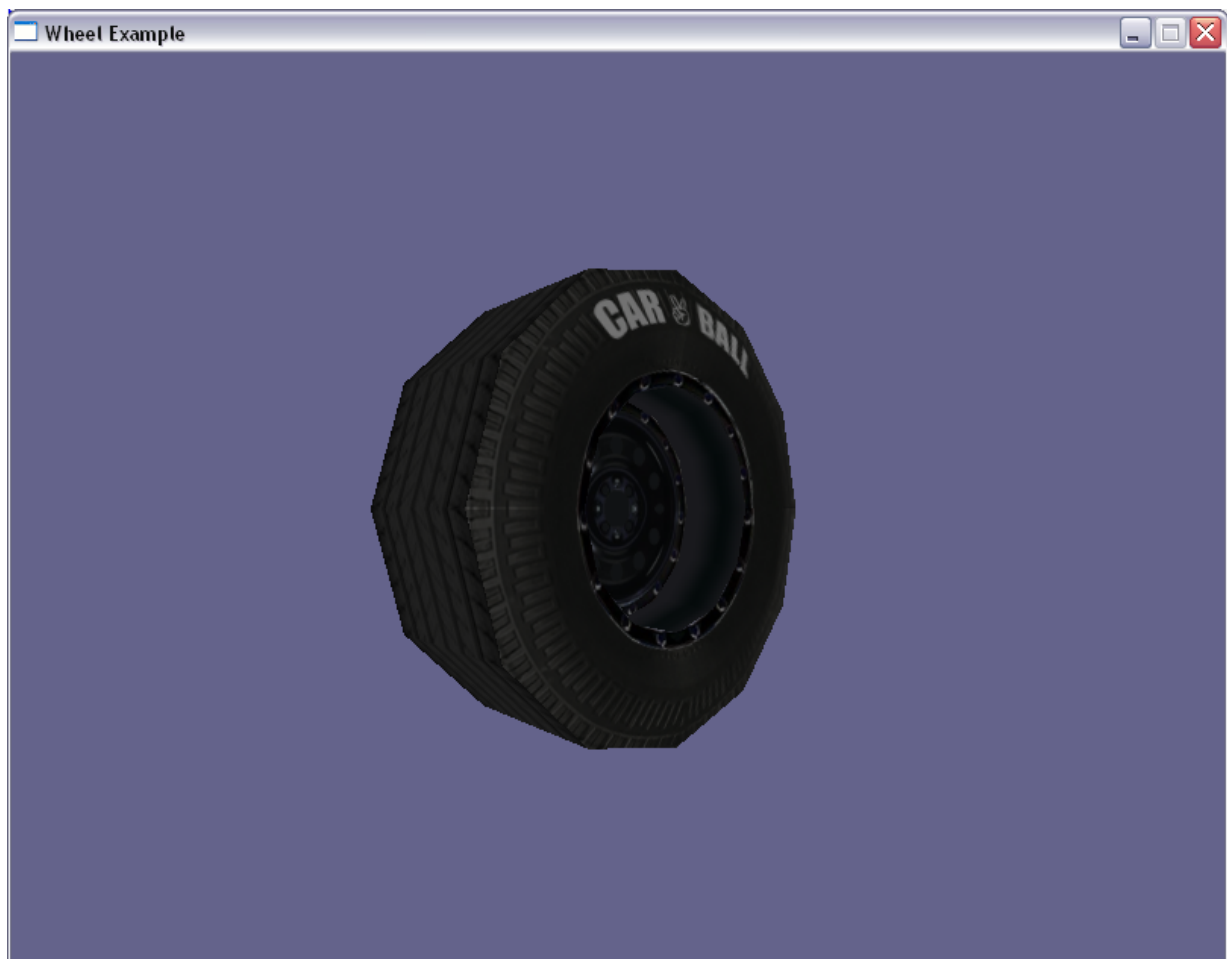
Using [PointEntity\(\)](#) allows us tell Blitz we want the camera to face the wheel, saving us time fiddling about with the rotation of the camera manually

We used the [CameraClsColor\(\)](#) command to change the background colour of the engine when rendering. Experiment with the colour values until you find a colour you like

Now we'll create the main loop and see how everything looks;

```
;-----  
;MAIN LOOP  
;-----  
  
While Not KeyHit(1)    ;if escape key hasn't been pressed  
  
    UpdateWorld  
    RenderWorld  
  
    Flip  
Wend  
  
End
```

Run the program and you should see something like this;



## Acceleration

Ok, the mesh loads and we can see it. Now we'll start with some basic movement. Before the main loop declare the following values;

```
Global speed# = 0           ;amount to spin the wheel by
Const accel# = 0.2         ;amount to increase the speed by
Const decel# = 0.01       ;amount to decrease the speed by
Const topspeed# = 35       ;speed at which to stop accelerating
```

The values should be pretty obvious. The `speed#` is the value used to spin the wheel, `accel#` is the value we'll increase the speed by to create acceleration, `decel#` is the value to decrease the speed by naturally (effectively friction) and `topspeed#` is the value we'll use to define when to stop accelerating and spin at a steady rate.

Again we tell the engine that `speed#` is global so we can access it outside of the main loop in the function we will use to rotate the wheel. The other variables are defined as `Const` or Constants because their values won't be changing within the program.

Notice the `#` after the variable name. Without this the program will not recognise that the value is a floating point variable (i.e. 1.002, 0.02, etc) and will treat it as an integer (a whole number such as 1,2 45, 639, etc). This will actually prevent the wheel from spinning but won't return an error.

Now we'll create the function that will allow us to move the wheel. We could add the code inside the main loop but it's much tidier to create a function and then tell Blitz to read that function inside the main loop.

At the end of the main loop (after 'End') move down a few lines and add the following;

```
Function SpinWheel()

    If KeyDown(200) ;If Up Arrow is held down - increase speed

        speed = speed + accel ;increase the speed by the accel amount
        If speed > topspeed# Then speed = topspeed# ;Prevent further
                                                    ;acceleration

    EndIf

    TurnEntity(wheel, speed, 0, 0)

End Function
```

Let's take a look at what's going on here. If we press the Up arrow, the `speed#` is increased gradually by the `accel#` variable. Then we say, if the current speed of the wheel reaches above `topspeed#` value, set the speed to that value to stop the wheel accelerating. We use 'greater than' as oppose to 'equals' to prevent the speed jumping from 99.99 to 100.01 and thus not acknowledging that we want it to stay at the top speed. Using 'greater than' ensures that the statement is always recognised. Basically we say, 'if the speed is above the top speed, set the speed to be the top speed'.

Meanwhile, the `TurnEntity()` command turns the wheel around the X axis by the speed value stated in the If/EndIf statement before it.

In order to tell the engine to read this function we simply put the name of it in the main loop. Before `'UpdateWorld'` add the following;

```
SpinWheel()
```

That's all that's needed to link the function into the main loop

Finally, in the main loop after 'RenderWorld' we'll add text to allow us to monitor the speed of the wheel;

```
Text 10,10,"Speed = "+speed ;place text on screen
```

Text is 2D and so only requires X and Y coordinates which in this case are 10 and 10. Adding '+speed' directly after the string (text) we want displayed in screen will display the current variable value on screen. This can be used to display ammo, health, number of enemies left, anything really. Run the program and check everything works. The wheel should start spinning slowly, getting faster until it reaches the top speed. Also the text should increase with the speed.

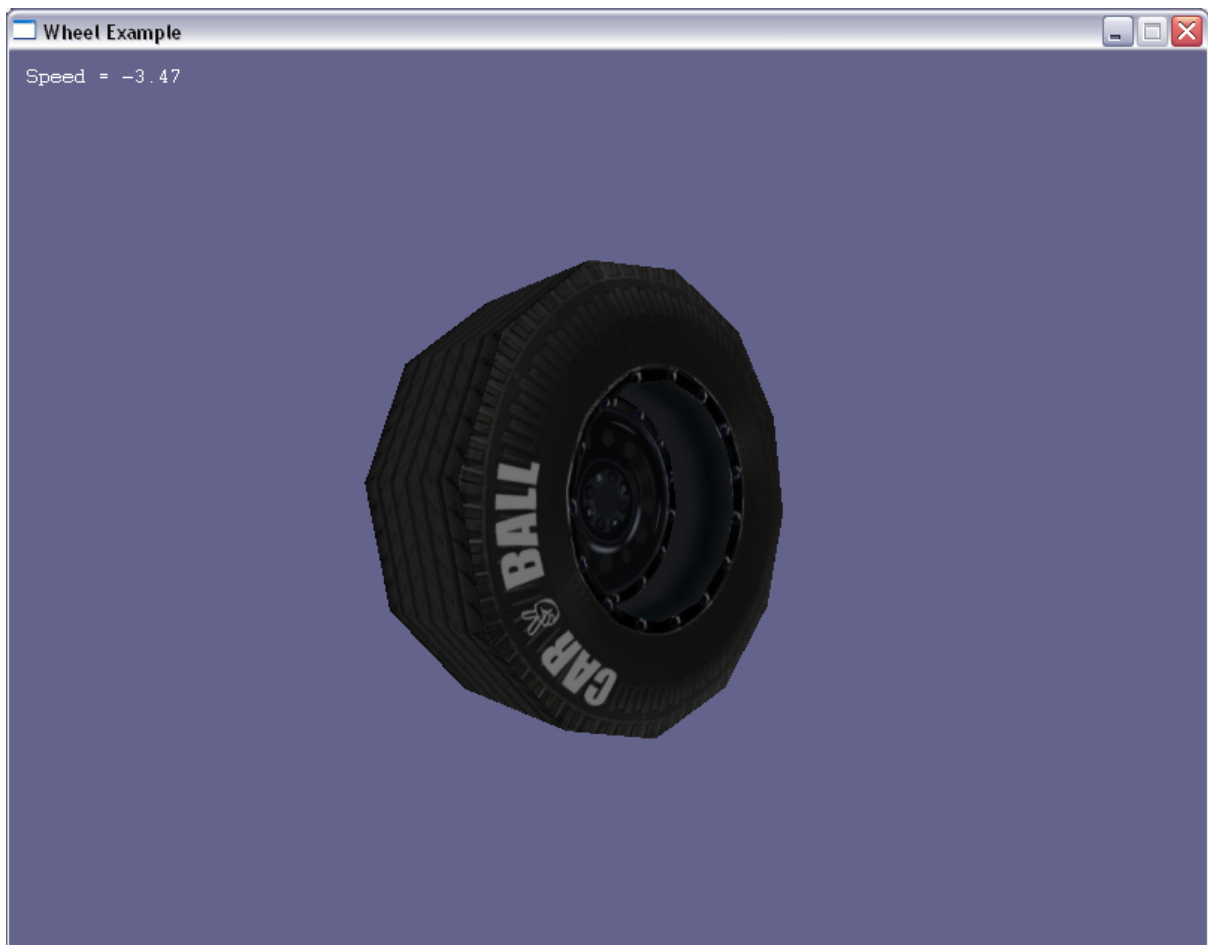
If you press the Up arrow the wheel will start spinning and stay spinning constantly, even if you only pressed it once. To counter this and slow the wheel down, we need to simulate gravity and gradually slow the wheel down until it stops.

### Deceleration

This is incredibly simple but can be a real pain to figure out at first (believe me). In the `SpinWheel()` function, before the If/EndIf statement, add the following;

```
speed = speed - decel ;reduce the speed
```

What we're doing here is telling the engine to repeatedly reduce the speed if the wheel, even when it is not spinning. Run the program and see what happens.



You'll notice that the wheel slows down after you've pressed the up arrow, but it continues spinning backwards and accelerating in reverse. We can fix this the same way we told the engine to stop the acceleration on the wheel when it reaches it's top speed.

Simply add the following to the `SpinWheel()` function after all the other statements in the function;

```
If speed<0 Then speed = 0
```

In the same way we prevented infinite acceleration, we tell Blitz 'If the speed goes below 0, set the speed to 0' stopping it spinning.

Run the program and take a look at the results. Hopefully the wheel should gradually slow to a stop after you've pressed and released the up arrow.

Finally we'll implement a breaking feature into the code to allow us to slow the wheel down. This is very simple. First add another `Const` variable at the start of the program to control the amount we want to break by;

```
Const break# = 0.3      ;amount to break by
```

Now in the `SpinWheel()` function add the following code snippet after the first `If/EndIf` statement;

```
If KeyDown(208) ;Down arrow - slow down

    speed = speed - break ;reduce the speed by the break amount
    If speed<0 Then speed = 0 ;stop the wheel when the speed is below 0

End If
```

Basically, we're telling Blitz to reduce the speed by the break variable when we press the down arrow. Then we again stop the wheel when the speed is below 0. If you do not add the 'If speed <0...' the wheel will eventually start turning backwards at the speed stated in the break variable.

That's it! Simple when you know how.

Original wheel model made by GamesterArt and is available as part of a great buggy model from

<http://www.turbosquid.com/FullPreview/Index.cfm/ID/273677>