BLITZ3D

9-Lesson

CRASH COURSE

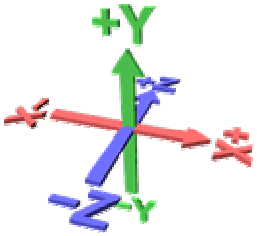
by

J-Man
(jnoodle.com)

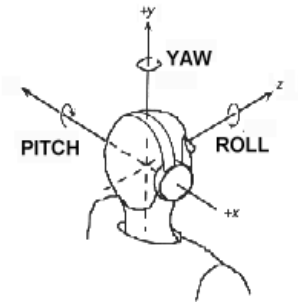
Table of Contents

1. Introduction

ORIENTING YOURSELF



Unlike the standard XY coordinate system of 2D games where the upper left corner of the screen is (0,0) , the 3D game world positions the XY axes on the CENTER of the screen. The Z-direction is oriented as forward/backward. This can be confusing to 2D coders since Z is typically an UP/DOWN vertical orientation scheme. When you place or move an object in your 3D world it is easy to see this XYZ relationship, BUT once you start moving a CAMERA things can get disorienting pretty quickly.



Placing an object in 3D space has more variety than just an x,y,z position. You can also TURN an object (camera included) about each of the X, Y & Z axes. Turning about the X is called PITCH, Y is YAW and Z is ROLL, just like if you were flying an airplane (see diagram). These turns are measured in degrees (0-360).

Before diving into your first program, please take a few moments to review the following coding terminology excerpted from a tutorial by Paul Gerfen. If you get confused, don't worry. The concepts will be introduced at the appropriate time throughout this tutorial so you can see how they work in action!

The Game Frame

The following code represents the general organizational framework of a common game. The yellow comments offer additional explanation and are ignored by Blitz because of the preceding semi-colon. Commenting your code is a prefessional practice and will save you hours of frustration down the road. If you remove all the comments you can appreciate how little code is actually required to get a simple 3D game up and going!

```
; Insert includes, declare global variables, dimension arrays

; Set video mode
Graphics3D 640,480,16,2
SetBuffer BackBuffer()
; Setup cameras
cam1 = CreateCamera()
CameraViewport cam1,0,0,640,480
; Load level objects, e.g. meshes, lights, terrain, water plane, skybox, set collisions...
light1=CreateLight()
cube1=CreateCube()
PositionEntity cube1,-2,-2,10
; The MAIN GAME LOOP
While Not KeyHit(1)
    ; Keyboard/Mouse Controls
    ; 3D Stuff, e.g. move/animate meshes, launch projectiles, check collisions...

    UpdateWorld
    RenderWorld

    ; 2D stuff here, update HUD, text, stats...
    Text 50,50,"Hello Cube!"

    Flip
Wend

End

; Subroutines & Functions can be here
```

NEW COMMANDS

```
; Comments always start with a semi-colon
Graphics3D 640,480,16,2
```

Graphics3D 640,480,16,2 sets the 3D graphics card to a resolution of 640x480 using 16 bit color and windowed (1=full screen, 2=windowed). You will find that running fullscreen can speed up your game significantly in many cases.

SetBuffer BackBuffer() *, goes with FLIP*

Using **SetBuffer BackBuffer()** is a technique in animation where you draw your world objects to a hidden back page (aka Backbuffer) while the frontpage is being shown. When the new world changes have been completed then the pages are switched (Flip). This prevents a problem known as screen flicker.

```
light1=CreateLight()
cam1 = CreateCamera()
CameraViewport cam1,0,0,640,480
```

light1=CreateLight() creates a standard level light named **light1**. Coloring, pointing, and ranging lights will be covered later.

cam1 = CreateCamera() creates a camera named **cam1**. You can control where this camera points in your world with the keyboard (covered later) creating a 1st Person view.

CameraViewport cam1,0,0,640,480 sets up the portion of the screen that will display what **cam1** "sees" (what is placed in front of it) and in this case it's the entire game screen. You can create multiple cameras for effects like split screen, rearview mirror, etc...

```
cube1=CreateCube()
PositionEntity cube1,-2,-2,10
```

cube=CreateCube() simply creates a cube in your world cenetered at 0,0,0 with opposite corners reaching from -1,-1,-1 to +1,+1,+1. This **primitive** shape **entity** can be **Scaled, RePositioned, Rotated, Colored, Textured and Alpha (transparency) adjusted** to suit a wide variety of world building purposes. e.g. EntityColor cube1, 255,0,0 will color the cube red. Other primitive entities you can create are **Planes, Spheres, Cylinders and Cones**. You can also add the number 16 or 32 in the parenthesis of you want a higher segmented (smoother) looking shape, e.g. CreateSphere(16).

While Not KeyHit(1) *, goes with Wend*

This begins our **MAIN GAME LOOP** and checks to see if the **Escape Key (scancode = 1)** has **NOT** been pressed **KeyHIT(1)**. If it has, the game **End**s.

The game loop consists of checking for additional key or mouse input from the user, giving each object a fair share opportunity to be moved a tad (player, enemies, projectiles) , checking for collisions, updating stats, and displaying the HUD GUI (Heads Up Display Graphical User Interface) e.g. score, life, mana, power, radar, etc.

3D objects must be adjusted first followed by the 2D elements since whatever is updated/drawn last can overwrite on top of what was drawn before. **Wend** designates the **end** of the **W**hile loop.

```
  UpdateWorld
  RenderWorld
```

UpdateWorld updates the objects/entities that we may have previously moved, scaled, rotated, etc. plus dealing with any collisions we might have set up outside the LOOP. **RenderWorld** graphically renders the objects/scene to the BackBuffer() (hidden page) awaiting the FLIP command to actually display to the screen. The **RenderWorld** command can also have an additional value put after this command to help slower computers increase their framerate. This is called **delta tweening** and won't be discussed in this tutorial because our lessons here won't deal with a huge world with tons of objects.

```
  Text 50,50,"Hello Cube"
  Flip
Wend

End
```

After **RenderWorld** we can now place our 2D elemenets, **e.g. score, life, mana, power, radar, etc.**

Text 50,50,"Hello Cube" places the words **"Hello Cube"** at X,Y game screen location **50,50** measured from the upper left corner. **Text**/**Sprites**/**Images** are placed/updated here before the **FLIP** command that then brings our hidden page (with BOTH 3D and 2D elements) to the screen. **Wend** sends us back to the start of the game loop for the next update of all our world object changes all over again.

THE CHALLENGE!!!

- **Create a scene with several different primitives repositioned to make a sweet scene.**

---

2. Control

In this lesson you will learn how to create reusable Functions that you can use to control movement of your player, or camera, or other characters/pieces. These functions can be created in a separate document and then Included in the main program to help keep things organized as well as reusable for other future programs! You will also be introduced to Parenting which permits you to "leash" one object to another, e.g. attach the camera to follow behind your character. And finally, you'll experience a bit of game logic and flow control with the IF..THEN and CASE commands.

CODE

Notice the **new code in bold** below that has been added to the previous lesson. Also notice below the **game2.bb** code **another separate page of code** called **gamefunctions2.bb**. These 2 separate pages are actually connected through the **Include "gamefunctions2.bb"** command.

game2.bb

```
Include "gamefunctions2.bb"

Graphics3D 640,480,16,2
SetBuffer BackBuffer()
cam1 = CreateCamera()
CameraViewport cam1,0,0,640,480
light1=CreateLight()
cube1=CreateCube()
PositionEntity cube1,0,0,10
sphere1=CreateSphere()
PositionEntity sphere1,0,0,20
;EntityParent cam1,cube1
While Not KeyHit(1)
    object_key_control( cam1 )
    random_move ( sphere1 )
    UpdateWorld
    RenderWorld
    Text 50,50,"Use arrow keys to control"
    Flip
Wend
End
```

gamefunctions2.bb

```
Function object_key_control( obj )
    If KeyDown( 200 )=True Then MoveEntity obj,0,0,1
    If KeyDown( 208 )=True Then MoveEntity obj,0,0,-1
    If KeyDown( 203 )=True Then TurnEntity obj,0,2,0
    If KeyDown( 205 )=True Then TurnEntity obj,0,-2,0
End Function
Function random_move ( obj )
    direction = Rand(1,2)
    MoveEntity obj,0,0,.5
    Select direction
        Case 1
            TurnEntity obj, 0, 10, 0
        Case 2
            TurnEntity obj, 0, -10, 0
    End Select
End Function
```

NEW COMMANDS

```
Include "gamefunctions2.bb"
...
object_key_control( cam1 )
```

Include "*gamefunctions.bb***"** is a powerful and simple command that allows you to insert big chunks of code from a separate document to keep things organized and compartmentalized for easy management.

```
Function object_key_control( obj )
    ...
End Function
```

These commands are in the separate **Include** file **gamefunctions.bb**. This function is "called" from the main **game.bb** program with the **object_key_control(obj)** code. Naming the function, i.e. **object_key_control(obj)**, is up to you and the wording should reflect what the function actually does. In this case, our function controls movement of an object with the keyboard. The object can be a camera, player, enemy, light, terrain, most anything! A function can even pass and/or return values through the use of a variable, i.e. (obj). You may wish to return more than one value or no value at all.

```
If KeyDown( 200 )=True Then MoveEntity obj,0,0,1
...
If KeyDown( 203 )=True Then TurnEntity obj,0,2,0
```

Within our first function **object_key_control(obj)** we combine a little logic **IF...THEN** with a check to see if a specific key is being pressed **KeyDown(200)=True** and if true, then we move the **obj**ect forward (+Z-direction) with **MoveEntity obj,0,0,1**. The numbers 200, 203... are **scancodes.** There is a specific scancode for each of the keyboards 100+ keys and can be gotten **here**. **If KeyDown(203)=True Then TurnEntity obj,0,2,0** will turn (YAW) the object LEFT when the **left arrow key** (203) is being pressed.

```
direction = Rand(1,2)
```

This little piece of code chooses a random **integer** number (number without a decimal) from 1 - 2 (1 or 2 actually). **x = Rand(-10,10)** would pick a random integer number from -10 up to +10. You might use **roll = Rand(1,20)** if you wanted to generate D20 stats for a role playing game.

```
Select direction
    Case 1
        TurnEntity obj, 0, 10, 0
    Case 2
        ...
End Select
```

Once you've generated a random number (1 or 2) and saved it into the variable **"direction**" you may wish to do something with each **SELECTION** possibility. With the **CASE** code we can offer the object the cance to randomly turn left i.e. **Case 1 ... (TurnEntity obj, 0, 10, 0)** or right i.e. **Case 2 ... (TurnEntity obj, 0, -10, 0)**.

THE CHALLENGE!!!

1. **Investigate what removing the ";" from the line ;EntityParent cam1,cube1**
2. **Change control keys and add a vertical up/down option**
3. **Try changing cam1 to cube1 and then light1 and explain what you see.**

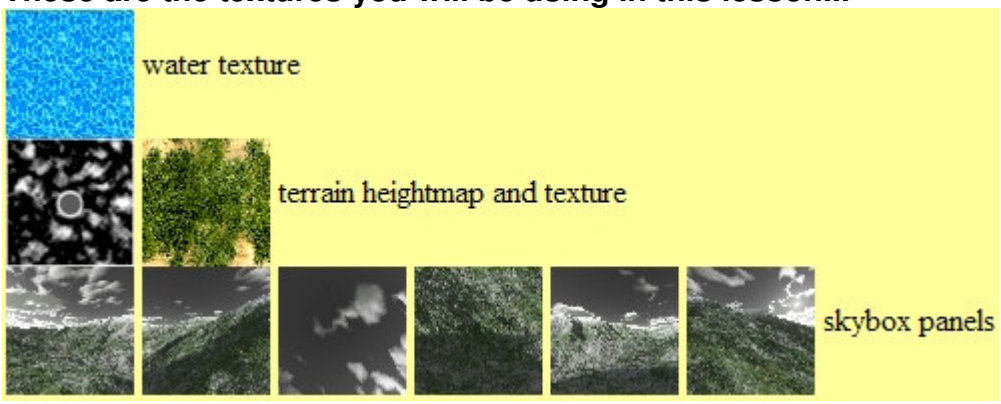
---

3. Water, Terrain & Sky

Water Plane, Terrain Mesh, Skyboxes & Texturing

Get ready to take a huge leap into 3D world creation! In this 3rd significant Waypoint Lesson you will (1.) create and apply a tiling/repeating water texture to a semi-transparent plane, (2.) use a heightmap image to generate a custom terrain applying a tiling/repeating ground texture and lastly (3.) grapple with the SKYBOX Beast by slapping a set of 6 "stiched" sky images onto a "hand-made" cube mesh. There are definitely easier ways of creating skies but this is the more common method with better looking results.

These are the textures you will be using in this lesson...



CODE

Notice the **new code in bold** below that has been added to the previous lesson. Also notice below the **game3.bb** code **another separate page of code** called **gamefunctions3.bb**.

game3.bb

```
Include "gamefunctions3.bb"
Global skybox, land
Graphics3D 640,480,16,2
SetBuffer BackBuffer()
cam1 = CreateCamera()
CameraViewport cam1,0,0,640,480
light1 = CreateLight()
sphere1 = CreateSphere()
EntityColor sphere1,255,0,0
ScaleEntity sphere1,10,10,10
MakeWater()
MakeTerrain()
```

```
MakeSkybox()
While Not KeyHit(1)
    object_key_control( cam1 )
    random_move ( sphere1 )

    PositionEntity skybox,EntityX(cam1),EntityY(cam1),EntityZ(cam1)
    UpdateWorld
    RenderWorld
    Text 50,50,"x="+EntityX(cam1)+" y="+EntityY(cam1)+" z="+EntityZ(cam1)
Flip
Wend
End
```

gamefunctions3.bb

```
Function MakeWater()
  water_tex=LoadTexture( "water.jpg" )
  ScaleTexture water_tex,20,20
  water=CreatePlane()
  EntityTexture water,water_tex
  PositionEntity water,0,4,0
  EntityAlpha water,.5
End Function
Function MakeTerrain()
  land_tex=LoadTexture( "terrain_tex.jpg" )
  ScaleTexture land_tex,10,10
  land=LoadTerrain( "terrain_hmap.jpg" )
  EntityTexture land,land_tex
  PositionEntity land,-2048,0,-2048
  ScaleEntity land,8,100,8
  TerrainDetail land,1000,True
End Function
Function MakeSkybox()
  skybox=CreateMesh()
  ;front face
  b=LoadBrush( "front.jpg",49 )
  s=CreateSurface( skybox,b )
  AddVertex s,-1,+1,-1,0,0:AddVertex s,+1,+1,-1,1,0
  AddVertex s,+1,-1,-1,1,1:AddVertex s,-1,-1,-1,0,1
  AddTriangle s,0,1,2:AddTriangle s,0,2,3
  FreeBrush b
  ;left face
  b=LoadBrush( "left.jpg",49 )
  s=CreateSurface( skybox,b )
  AddVertex s,+1,+1,-1,0,0:AddVertex s,+1,+1,+1,1,0
  AddVertex s,+1,-1,+1,1,1:AddVertex s,+1,-1,-1,0,1
```

```
  AddTriangle s,0,1,2:AddTriangle s,0,2,3
  FreeBrush b
  ;back face
  b=LoadBrush( "back.jpg",49 )
  s=CreateSurface( skybox,b )
  AddVertex s,+1,+1,+1,0,0:AddVertex s,-1,+1,+1,1,0
  AddVertex s,-1,-1,+1,1,1:AddVertex s,+1,-1,+1,0,1
  AddTriangle s,0,1,2:AddTriangle s,0,2,3
  FreeBrush b
  ;right face
  b=LoadBrush( "right.jpg",49 )
  s=CreateSurface( skybox,b )
  AddVertex s,-1,+1,+1,0,0:AddVertex s,-1,+1,-1,1,0
  AddVertex s,-1,-1,-1,1,1:AddVertex s,-1,-1,+1,0,1
  AddTriangle s,0,1,2:AddTriangle s,0,2,3
  FreeBrush b
  ;top face
  b=LoadBrush( "up.jpg",49 )
  s=CreateSurface( skybox,b )
  AddVertex s,-1,+1,+1,0,1:AddVertex s,+1,+1,+1,0,0
  AddVertex s,+1,+1,-1,1,0:AddVertex s,-1,+1,-1,1,1
  AddTriangle s,0,1,2:AddTriangle s,0,2,3
  FreeBrush b
  ;bottom face
  b=LoadBrush( "down.jpg",49 )
  s=CreateSurface( skybox,b )
  AddVertex s,-1,-1,-1,1,0:AddVertex s,+1,-1,-1,1,1
  AddVertex s,+1,-1,+1,0,1:AddVertex s,-1,-1,+1,0,0
  AddTriangle s,0,1,2:AddTriangle s,0,2,3
  FreeBrush b
  ScaleMesh skybox,100,100,100
  FlipMesh skybox
  EntityFX skybox,1
  EntityOrder skybox,10
End Function
Function object_key_control( obj )
  If KeyDown( 200 )=True Then MoveEntity obj,0,0,1
  If KeyDown( 208 )=True Then MoveEntity obj,0,0,-1
  If KeyDown( 203 )=True Then TurnEntity obj,0,2,0
  If KeyDown( 205 )=True Then TurnEntity obj,0,-2,0
  ex#=EntityX(obj):ez#=EntityZ(obj)
  PositionEntity obj,ex,TerrainY( land,ex,0,ez )+5,ez
End Function
Function random_move ( obj )
  direction = Rand(1,2)
  MoveEntity obj,0,0,.5
```

```
  Select direction
    Case 1
      TurnEntity obj, 0, 10, 0
    Case 2
      TurnEntity obj, 0, -10, 0
  End Select
  ex#=EntityX(obj):ez#=EntityZ(obj)
  PositionEntity obj,ex,TerrainY( land,ex,0,ez ),ez
End Function
```

NEW COMMANDS

```
Global skybox, land
...
MakeWater()
MakeTerrain()
MakeSkybox()
...
    PositionEntity skybox,EntityX(cam1),EntityY(cam1),EntityZ(cam1)
...
    Text 50,50,"x="+EntityX(cam1)+" y="+EntityY(cam1)+" z="+EntityZ(cam1)
```

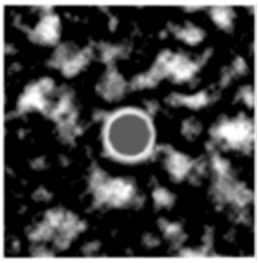
Global skybox, land declares our terrain object **land** and skybox object **skybox** to be globally accessible. Some languages refer to this as **making them public**. When we use functions to create the skybox and terrain i.e. **MakeTerrain()** and **MakeSkybox()**, we often will want to manipulate or access properties of the skybox and terrain AFTER we create them. A skybox requires that the box be attached to the player's view, otherwise you wil quickly reach the end of the sky :)

PositionEntity skybox,EntityX(cam1),EntityY(cam1),EntityZ(cam1)
takes care of this. **EntityX(cam1)** is a Blitz function that obtains the x-coordinate of the enclosed object, in this case **cam1**. The skybox is then constantly repositioned in the GAMELOOP on the player so that you never can catch up to the sky (try commenting out this line to see the strange effect). Using this same **EntityX(cam1)** in the following **Text** command will update our HUD with our X,Y,Z position.

```
Function MakeWater()
  water_tex=LoadTexture( "water.jpg" )
  ScaleTexture water_tex,20,20
  water=CreatePlane()
  EntityTexture water,water_tex
  PositionEntity water,0,4,0
  EntityAlpha water,.5
End Function
```

Function MakeWater() is fairly straightfoward. **water_tex=LoadTexture("water.jpg")** loads the image **water.jpg** into the variable **water_tex**. **ScaleTexture water_tex,20,20** then enlarges this texture by a factor of 20 times. **water=CreatePlane()** creates an "infinite" plane with the handle **water**. **EntityTexture water,water_tex** then applies our tiling water image to the water plane. After positioning it, we make the water plane 50% (.5) transparent with the **EntityAlpha water,.5** command.

```
Function MakeTerrain()
  land_tex=LoadTexture( "terrain_tex.jpg" )
  ScaleTexture land_tex,10,10
  land=LoadTerrain( "terrain_hmap.jpg" )
  ScaleEntity land,8,100,8
  PositionEntity land,-2048,0,-2048
  EntityTexture land,land_tex
  TerrainDetail land,1000,True
End Function
```



Making a terrain is a little more involved. Blitz has a really cool function that will allow you to take a grayscaled image **"terrain_hmap.jpg"** (preferably 256x256 or **512x512** pixels) and use the color variation to adjust the relative height of the vertexes of a terrain mesh . Since our heightmap is 512x512 pixels which **LoadTerrain("terrain_hmap.jpg")** translates to 512x512 world units, **ScaleEntity land,8,100,8** scales the horizontal dimensions to 4096 (512 x 8) units and the vertical units to 100. To center this huge terrain about the 0,0,0 world origin we use **PositionEntity land,-2048,0,-2048** (2048 is 1/2 of 4096). **TerrainDetail land,1000,True** is not a critical Blitz3D command but it does improve the rendering of large terrains more efficiently on slower computers. Increase this number for more detail and decrease it for increased speed.

```
Function MakeSkybox()
  skybox=CreateMesh()
  ;front face
  b=LoadBrush( "front.jpg",49 )
  s=CreateSurface( skybox,b )
  AddVertex s,-1,+1,-1,0,0:AddVertex s,+1,+1,-1,1,0
  AddVertex s,+1,-1,-1,1,1:AddVertex s,-1,-1,-1,0,1
  AddTriangle s,0,1,2:AddTriangle s,0,2,3
  FreeBrush b
    . . . repeat with remaining 5 faces of skybox cube
  ScaleMesh skybox,100,100,100
  FlipMesh skybox
  EntityFX skybox,1
  EntityOrder skybox,10
End Function
```

Whether you understand this piece of skybox code or not is OK, this code will be the same for all your future skyboxes and is completely reusable, **This chunk of SKYBOX CREATION code is HUGE** compared to what you've done previously but if you look carefully you will see a repeating pattern within the function. A skybox, like any box, has 6 sides and making a skybox requires us to manually create, vertex by vertex, triangle by triangle (2 per side), each of the six sides and then texture each side with a unique "stitched" sky image. Many sites offer free skybox texture sets (dnainternet.net or .3delyvisions.com). After all the sides have been created and the textures applied, we have to use the **FlipMesh skybox** command to turn our box outside-in, because our player/camera will be placed on the inside of the skybox box. **EntityOrder skybox,10** is another trick that just makes our level look better where the terrain horizon meets our skybox.

```
ex#=EntityX(obj):ez#=EntityZ(obj)
PositionEntity obj,ex,TerrainY( land,ex,0,ez ),ez
```

In our previous control functions

Function object_key_control(obj) and **Function random_move (obj)**
we didn't have to worry about our vertical direction (Y), but now that we have a terrain we need our objects to "hug" the terrain when they travel. Otherwise, we will travel right through the terrain. **TerrainY(land, x, 0, z)** is a great Blitz3D function that finds the vertical Y- value of the terrain wherever an object is located on the horizontal (X-Z).

THE BIG 3 CHALLENGE!!!

- **Change the terrain by modifying the monochromatic heightmap terrain_hmap.jpg texture using a gradient spraypaint tool in a good paint program**
- **Find a different tiling texture (terrain_tex.jpg) to apply to the terrain.**
- **Locate and implement a new skybox (set of 6 "stitched" images). (dnainternet.net or .3delyvisions.com)**
- **Demonstrate to the instructor when ready :)**

---

4. Meshes Static/Animated

3D Models - Static and Animated

Your world can be filled with static meshes or animated meshes. Static meshes are usually trees, rocks, furniture, buildings, weapons, and various pickups. Animated meshes are usually moving characters, animals, etc. In this tutorial you will be adding trees to your landscape and getting chased by dogs! Don't worry, they won't bite.....yet. You will be making a few additional modifications to make your code more flexible and efficient. Finally you will be introduced to the "For ... Next" loop and the mighty TYPES convention, both allowing you to create and manage hordes of objects and creatures! In this demo you will add 50 independent animated dogs and 1000 trees!

---

entity = LoadMesh (file$) loads a **STATIC** mesh from a .X, .3DS or .B3D file
entity = LoadAnimMesh (file$) loads an **ANIMATED** mesh from a .X, .3DS, .B3D or .MD2 file**.** After loading the animated mesh the individual animations must be located, extracted and assigned a unique number by **ExtractAnimSeq(entity, first_frame, last_frame)**. Once extracted the entity animation can be activated with **Animate entity[,mode][,speed#][,sequence][,transition#]**

from the Blitz3D Help:
mode *(optional) - mode of animation.*
0: stop animation 1: loop animation (default) 2: ping-pong animation 3: one-shot animation
speed# *(optional) - speed of animation. Defaults to 1. - a negative speed will play the animation backwards.*
sequence *(optional) - specifies which sequence of animation frames to play. Defaults to 0.*
transition# *(optional) - used to tween between an entities current position rotation and the first frame of animation. Defaults to 0. A value of 0 will cause an instant 'leap' to the first frame, while values greater than 0 will cause a smooth transition.*

game4.bb

```
Include "gamefunctions4.bb"
Global skybox, land, cam1
Graphics3D 640,480,16,2
SetBuffer BackBuffer()
cam1 = CreateCamera()
CameraViewport cam1,0,0,640,480
light1 = CreateLight()
waterlevel=4
MakeWater("water.jpg",waterlevel)
MakeTerrain()
MakeSkybox()
LoadTrees("tree.b3d")
Type dog
```

```
  Field ID
End Type
LoadDogs("dog.b3d")
While Not KeyHit(1)
  object_key_control( cam1 )
  MoveDogs()
  PositionEntity skybox,EntityX(cam1),EntityY(cam1),EntityZ(cam1)

  UpdateWorld
  RenderWorld

  ;2D stuff here
  Flip
Wend
End
```

partial gamefunctions4.bb

```
; these are just the new and modified functions

Function LoadTrees(file$)
  tree=LoadMesh(file$)
  ScaleMesh tree,.1,.1,.1
  For n=1 To 1000
    tree_copy=CopyEntity(tree)
    ScaleEntity tree_copy,Rand(1,4),Rand(1,4),Rand(1,4)
    x=Rand(-2000,2000) :z=Rand(-2000,2000)
    PositionEntity tree_copy, x,TerrainY(land,x,0,z)-15,z
    RotateEntity tree_copy,0,Rand(1,180),0
    EntityAutoFade tree_copy,500,1000
  Next
  FreeEntity tree
End Function
Function LoadDogs(file$)
  model=LoadAnimMesh(file$)
  ExtractAnimSeq model,2,14
  ScaleEntity model,.2,.2,.2
  For c=1 To 50
    Pack.dog = New dog
    Pack\ID = CopyEntity(model)
    x=Rand(-2000,2000):z=Rand(-2000,2000)
    PositionEntity Pack\ID, x,TerrainY(land,x,0,z),z
    Animate Pack\ID,1,.3,1
    EntityAutoFade Pack\ID,500,1000
  Next
  FreeEntity model
```

```
End Function
Function MoveDogs()
  For moveall.dog = Each dog
    If EntityDistance(moveall\ID,cam1)>10
    PointEntity moveall\ID,cam1
    MoveEntity moveall\ID,0,0,.5
    ex#=EntityX(moveall\ID):ez#=EntityZ(moveall\ID)
    PositionEntity moveall\ID,ex,TerrainY( land,ex,0,ez ),ez
    EndIf
  Next
End Function

; can set different textures and water level now
Function MakeWater(file$,wl)
  water_tex=LoadTexture(file$)
  ScaleTexture water_tex,20,20
  water=CreatePlane()
  EntityTexture water,water_tex
  PositionEntity water,0,wl,0
  EntityAlpha water,.5
End Function
```

NEW COMMANDS

```
Global skybox, land, cam1
. . .
waterlevel=4
MakeWater("water.jpg",waterlevel)
. . .
LoadTrees("tree.b3d")
. . .
Type dog
  Field ID
End Type
LoadDogs("dog.b3d")
. . .
  MoveDogs()
```

We have to add **cam1** to the Global variable list so that the camera is accessible in our MoveDogs() function (The dogs are "pointed" to run toward you, the camera). We also modify our MakeWater() rountine **MakeWater("water.jpg",waterlevel)** so that we can now pass the function a water texture of our choice and adjust the water level too (**waterlevel=4**).

TYPES!!!

TYPE is power! It is used to create a group of objects that have the same characteristics that can be managed quickly and easily. Here we have created a dog type with **Type dog** and created a characteristic variable called ID with **Field ID**. We could also create additional variables like x-position, y-position, speed, texture, size, aggressiveness.... but for now ID will store the unique Blitz3D "handle" of each of the dogs we create in **LoadDogs("dog.b3d")**. **MoveDogs()** will then manage the movement of each dog by pointing it toward us and moving it.

In gamefunctions.bb...

```
Function LoadTrees(file$)
  tree=LoadMesh(file$)
  ScaleMesh tree,.1,.1,.1
  For n=1 To 1000
    tree_copy=CopyEntity(tree)
    ScaleEntity tree_copy,Rand(1,4),Rand(1,4),Rand(1,4)
    x=Rand(-2000,2000) :z=Rand(-2000,2000)
    PositionEntity tree_copy, x,TerrainY(land,x,0,z)-15,z
    RotateEntity tree_copy,0,Rand(1,180),0
    EntityAutoFade tree_copy,500,1000
  Next
  FreeEntity tree
End Function
```

Function LoadTrees(file$) receives the passed model filename through **file$** and then loads the mesh into **tree=LoadMesh(file$)**. Because our tree model is so gigantic we reduce the mesh to 10% of it's original size with **ScaleMesh tree, .1, .1, .1**

For n=1 To 1000 . . . Next is a way to iterate (cycle, loop, repeat multiple times) through our **tree_copy=CopyEntity(tree)** making 1000 trees. After each tree is copied we use a technique to randomly **scale ScaleEntity tree_copy, Rand(1,4), Rand(1,4), Rand(1,4)** , randomly **position** (**x=Rand(-2000,2000) :z=Rand(-2000,2000) . . . PositionEntity tree_copy, x,TerrainY(land,x,0,z)-15,z**) , and randomly **rotate RotateEntity tree_copy,0,Rand(1,180),0** each tree copy to **create the illusion that we have a forest of 1000 unique trees**. To keep our huge world from slowing with so many objects, we employ **EntityAutoFade tree_copy,500,1000** which causes anything 1000 units away from our camera to fade away. Fading starts at 500 with complete invisibility at 1000 units. We can then use **FreeEntity tree** to free up some memory by dumping our original **tree**. **Ideally, when your game ends, or you load the next level, ALL entities should be freed up from memory.**

```
Function LoadDogs(file$)
  model=LoadAnimMesh(file$)
  ExtractAnimSeq model,2,14
  ScaleEntity model,.2,.2,.2
  For c=1 To 50
    Pack.dog = New dog
    Pack\ID = CopyEntity(model)
    x=Rand(-2000,2000):z=Rand(-2000,2000)
    PositionEntity Pack\ID, x, TerrainY(land,x,0,z), z
    Animate Pack\ID,1,.3,1
    EntityAutoFade Pack\ID,500,1000
  Next
  FreeEntity model
End Function
```

Function LoadDogs(file$) receives the passed animated model filename through **file$** and then loads the animated mesh into **model=LoadAnimMesh(file$)**.
As mentioned in the introduction, **ExtractAnimSeq model, 2, 14** extracts animation sequence frames 2-14 (running) as the animation sequence #1. If you extracted another sequence it would be autonumbered as animation sequence #2, etc. The dog models are fairly large for our world so we scale them to 20% with **ScaleEntity model, .2, .2, .2** .
TYPES again!!! TYPES will probably be difficult at first to wrap your brain around, and you could do without them if your game stays small scale, but true power awaits the coder that wields understanding of the TYPE. Remember in the main program we created our dog type with

```
Type dog
  Field ID
End Type
```

Now we get to use the power of TYPES to create a **Pack** of 50 dogs! Using a **For . . . Next** loop we iterate through 50 instances of **Pack.dog = New dog** which creates and sets each dog's **ID** handle with **Pack\ID = CopyEntity(model)** . **Pack** is our group name for the **dog** entities. Each dog is then randomly positioned on the terrain and autofaded. **Animate Pack\ID, 1, .3, 1** starts the animation **looping mode 1** for each dog running at 30% speed (**.3**) playing **sequence #1** (frames 2-14, extracted previously). After all 50 dogs are generated the original model is freed from memory.

```
Function MoveDogs()
  For moveall.dog = Each dog
    If EntityDistance(moveall\ID,cam1)>10
      PointEntity moveall\ID,cam1
      MoveEntity moveall\ID,0,0,.5
      ex#=EntityX(moveall\ID):ez#=EntityZ(moveall\ID)
```

```
      PositionEntity moveall\ID,ex,TerrainY( land,ex,0,ez ),ez
    EndIf
  Next
End Function
```

In our **Function MoveDogs()** we use a special **For . . . Each** that is designed specifically for **TYPES**.

For moveall.dog = Each dog . . . **Next** lets us loop through and make adjustments to **Each** instance of **dog** using the variable **moveall**

If EntityDistance(moveall\ID,cam1)>10 checks the distance between you **cam1** and each dog instance **moveall\ID** and if the dog is greater than 10 units away it will point it toward you **PointEntity moveall\ID, cam1** move it closer **MoveEntity moveall\ID,0,0,.5** and then make sure it stays on top of the terrain with **ex#=EntityX(moveall\ID): ez#=EntityZ(moveall\ID)** and **PositionEntity moveall\ID, ex, TerrainY(land,ex,0,ez), ez**

```
; can set different textures and water level
Function MakeWater(file$,wl)
  water_tex=LoadTexture(file$)
  . . .
  PositionEntity water,0,wl,0
  . . .
```

With these modifications to our **Function MakeWater(file$,wl)** we can now more easily pass different textures AND set the water level. Using a water level variable will come in handy in the next lesson when we add a water wading sound as the player enters the water.

THE CHALLENGE!!!

- **Modify dog & tree textures, for a unique level interpretation.**
- **Determine the maximum number of trees and dogs obtainable, balancing with EntityAutoFade, before significant slowdown is noticed**

---

5. Lighting, Sound

Time to spice up the game with some cool lighting, music and sound effects! For lighting, Blitz allows us to set the world ambient light level and color, create individual point and spotlight sources that can be positioned, colored, rotated and ranged. We are also going to introduce the new concept of READing an external included level DATA file into the game. This DATA file will contain the information for each lights x position, z position, Red, Green, Blue, and Range. This data file could also be loaded with information on everything else in your world, including skybox, terrain, sound, characters, etc. Blitz supports a wide range of music and sound formats, including raw, wav, mp3, and ogg. You will be adding some background music, a running sound and a wading sound for the water.

game5.bb

```
Include "gamefunctions5.bb"
Global skybox, land, cam1, wade_vol#, run_vol#, wadechannel, runchannel
Global waterlevel=4

Graphics3D 640,480,16,2
SetBuffer BackBuffer()

cam1 = CreateCamera()
CameraViewport cam1,0,0,640,480

MakeWater("water.jpg",waterlevel)
MakeTerrain()
MakeSkybox()
LoadTrees("tree.b3d")
LoadMusic("spacepanic.mp3")
LoadSFX()
AmbientLight 5,5,5
LoadLights()

Type dog
   Field ID
End Type
LoadDogs("dog.b3d")

While Not KeyHit(1)
   object_key_control( cam1 )
```

```
  ChannelVolume wadeChannel, wade_vol#
  ChannelVolume runChannel, run_vol#
  MoveDogs()
  PositionEntity skybox,EntityX(cam1),EntityY(cam1),EntityZ(cam1)

  UpdateWorld
  RenderWorld

  ;2D stuff here
  Flip
Wend
End

Include "game5data.bb"
```

This is the included gamefunctions file gamefunctions5.bb

```
Function LoadLights()
  Read numlights
  For n=1 To numlights
    Read x,z,R,G,B,range
    light=CreateLight(2) ;2=point light
    LightColor light, R,G,B
    LightRange light, range
    PositionEntity light, x,TerrainY(land,x,0,z)+50,z
  Next
End Function

Function LoadMusic(files$)
  bgsound = LoadSound (files$)
  LoopSound bgsound
  SoundVolume bgsound,.5
  PlaySound bgsound
End Function

Function LoadSFX()
  run = LoadSound ("gravel.wav")
  wade = LoadSound ("water.wav")
  SoundVolume run, 0
  SoundVolume wade, 0
  LoopSound run
  LoopSound wade
  runChannel = PlaySound (run)
  wadeChannel = PlaySound (wade)
End Function
```

```
Function object_key_control( obj )
  walksnd=False:wadesnd=False
  If KeyDown( 200 )=True And EntityY(obj)>waterlevel+4 Then
    MoveEntity obj,0,0,1:walksnd=True
  If KeyDown( 208 )=True And EntityY(obj)>waterlevel+4 Then
    MoveEntity obj,0,0,-1:walksnd=True
  If KeyDown( 200 )=True And EntityY(obj)<=waterlevel+4 Then
    MoveEntity obj,0,0,1:wadesnd=True
  If KeyDown( 208 )=True And EntityY(obj)<=waterlevel+4 Then
    MoveEntity obj,0,0,-1:wadesnd=True
  If KeyDown( 203 )=True Then TurnEntity obj,0,2,0
  If KeyDown( 205 )=True Then TurnEntity obj,0,-2,0
  ex#=EntityX(obj):ez#=EntityZ(obj)
  PositionEntity obj,ex,TerrainY( land,ex,0,ez )+5,ez
  If walksnd Then run_vol#=1 Else run_vol#=0
  If wadesnd Then wade_vol#=1 Else wade_vol#=0
End Function
```

This is the included data file game5data.bb

```
; This is an included file that contains your level data
; In this lesson it contains LIGHT data
; x,z, R,G,B, range
Data 4    ;number of lights
Data -200,0,    255,0,0,       100       ;red
Data 0,200,    0,255,0,       100       ;green
Data 200,0,    0,0,255,       100       ;blue
Data 0,-200,    255,255,0,    100    ;yellow
```

NEW COMMANDS

```
Global skybox, land, cam1, wade_vol#, run_vol#, wadechannel, runchannel
Global waterlevel=4
. . .
LoadMusic("spacepanic.mp3")
LoadSFX()
AmbientLight 5,5,5
LoadLights()
. . .
  ChannelVolume wadeChannel, wade_vol#
  ChannelVolume runChannel, run_vol#
. . .
Include "game5data.bb"
```

We first need to make some variables global, **wade_vol#, run_vol#, wadechannel, runchannel** will be used to control the volume of our 2 running sound effects. Since we need to change our running sound to a water wading sound when the player goes below the water level we need to globalize **waterlevel=4** variable.

LoadMusic("spacepanic.mp3") is our special function that loads our ambient mp3 background music**.**

LoadSFX() is our special function that loads our 2 sound effects.

AmbientLight 5, 5, 5 adjusts our world lighting to a dark gray so that our other lights stand out**.**

LoadLights() is our special function that loads our 4 custom point lights.

ChannelVolume wadeChannel, wade_vol# and **ChannelVolume runChannel, run_vol#** are within the **main game loop** and continuously updates our sound effects volume, e.g. wade sound volume is zero when we are running on dry land with our running sound at maximum. Since the volume is a value from 0-1 we need to have our variables decimal **#** rather than integer.

Include "game5data.bb" contains the position, color and range of the 4 point lights you'll see in the level that are READ by the **LoadLights()** function .

The following are from the included gamefunctions file gamefunctions5.bb

```
Function LoadLights()
  Read numlights
  For n=1 To numlights
    Read x,z,R,G,B,range
    light=CreateLight(2) ;2=point light
    LightColor light, R,G,B
    LightRange light, range
    PositionEntity light, x,TerrainY(land,x,0,z)+50,z
  Next
End Function
```

LoadLights() READS a line from the included data file (in sequence). **Read numlights** reads the first value "4" and uses this to loop (For...Next) **Read x,z,R,G,B,range** 4 times extracting the lights postion coordinates, color values and range. **light=CreateLight(2)** creates the point light, **LightColor light, R,G,B** sets the light color, **LightRange light, range** sets the range (default is 1000 units if ignored) and finally the light is positioned 50 units above the terrain.

```
Function LoadMusic(files$)
  bgsound = LoadSound (files$)
  LoopSound bgsound
  SoundVolume bgsound, .5
  PlaySound bgsound
End Function
```

LoadMusic(files$) receives the name of a music sound file and loads it into **bgsound =**

LoadSound (files$). We can then loop the music with **LoopSound bgsound** so that it repeats if we want. **SoundVolume bgsound, .5** obviously adjusts the music track volume to 50%. **PlaySound bgsound** starts the music playing. Simple!

```
Function LoadSFX()
  run = LoadSound ("gravel.wav")
  wade = LoadSound ("water.wav")
  SoundVolume run, 0
  SoundVolume wade, 0
  LoopSound run
  LoopSound wade
  runChannel = PlaySound (run)
  wadeChannel = PlaySound (wade)
End Function
```

Sound effects are a little trickier. The prefered method is to **load** each sound, set its starting **volume** to zero, **loop** it and assign it a **channel**. Once this is setup then the appropriate sound can be instantly accessed by just raising the volume when appropriate.

```
Function object_key_control( obj )
  walksnd=False:wadesnd=False
  If KeyDown( 200 )=True And EntityY(obj)>waterlevel+4 Then
    MoveEntity obj,0,0,1:walksnd=True
  If KeyDown( 208 )=True And EntityY(obj)>waterlevel+4 Then
    MoveEntity obj,0,0,-1:walksnd=True
  If KeyDown( 200 )=True And EntityY(obj)<=waterlevel+4 Then
    MoveEntity obj,0,0,1:wadesnd=True
  If KeyDown( 208 )=True And EntityY(obj)<=waterlevel+4 Then
    MoveEntity obj,0,0,-1:wadesnd=True
  If KeyDown( 203 )=True Then TurnEntity obj,0,2,0
  If KeyDown( 205 )=True Then TurnEntity obj,0,-2,0
  ex#=EntityX(obj):ez#=EntityZ(obj)
  PositionEntity obj,ex,TerrainY( land,ex,0,ez )+5,ez
  If walksnd Then run_vol#=1 Else run_vol#=0
```

```
   If wadesnd Then wade_vol#=1 Else wade_vol#=0
End Function
```

We will modify the player control routine so that the sound effects play at the appropriate times.

walksnd=False:wadesnd=False are special "flag" variables that we can toggle when we are **on land EntityY(obj)>waterlevel+4** or **in water EntityY(obj)<=waterlevel+4**.

The following is from the included data file game5data.bb

```
; This is an included file that contains your level data
; In this lesson it contains LIGHT data
; x,z, R,G,B, range
Data 4    ;number of lights
Data -200,0,    255,0,0,       100       ;red
Data 0,200,    0,255,0,       100       ;green
Data 200,0,     0,0,255,       100       ;blue
Data 0,-200,    255,255,0,    100    ;yellow
```

Here is the included LIGHT DATA that was refered to previously. **RGB** values can be obtained from any paint program in the color selection option

---

THE CHALLENGE!!!

- **Change the lights by modifying the DATA include, adjust the ambient lighting, and change the music for a unique level.**

---

6. Collisions

Collisions are the icing on interactivity. You can control your character all you want or launch a projectile, but if your characters can walk through trees or each other or your projectiles pass through objects like a ghost, a game can become quite boring and unrealistic. Collisions to the rescue! Collisions are remarkably easy to setup. Every different kind of object that you wish to have a collision reaction with must have their own unique ID number that is then assigned to them with the EntityType command (NOT the same as our previous definition of TYPE). Then the collisions between the different types are setup with the Collisions command. Each object can also be given a collision volume using the EntityRadius command.

game6.bb modifications

```
Global coll_player=1, coll_dogs=2, coll_trees=3
. . .

; Set collision type values . . . 2=slide by
Collisions coll_dogs,coll_dogs,1,2    ;dogs can't walk through dogs
Collisions coll_dogs,coll_trees,1,2   ;dogs can't walk through trees
Collisions coll_dogs,coll_player,1,2    ;dogs can't walk through you
Collisions coll_player,coll_trees,1,2   ;you can't walk through trees
Collisions coll_player,coll_dogs,1,2   ;you can't walk through dogs
EntityType cam1,coll_player
EntityRadius cam1,5
. . .
AmbientLight 50,50,50 ;up the light a tad
. . .
```

gamefunctions6.bb modifications

```
Function LoadTrees(file$)
. . .
  EntityRadius tree_copy, 10
  EntityType tree_copy,coll_trees
. . .
End Function
Function LoadDogs(file$)
. . .
  EntityRadius Pack\ID, 5
  EntityType Pack\ID,coll_dogs
. . .
```

```
End Function
;remove previous distance limit now that dogs have collisions
Function MoveDogs(obj)
  For moveall.dog = Each dog
   ;If EntityDistance(moveall\ID,obj)>10
      PointEntity moveall\ID,obj
      MoveEntity moveall\ID,0,0,.5
      ex#=EntityX(moveall\ID):ez#=EntityZ(moveall\ID)
      PositionEntity moveall\ID,ex,TerrainY( land,ex,0,ez ),ez
   ;EndIf
  Next
End Function
```

NEW COMMANDS

```
Global coll_player=1, coll_dogs=2, coll_trees=3
```

Need to globalize the object collision ID's so they can be accessed within the functions.

```
; Set collision type values . . . 2=slide by
Collisions coll_dogs,coll_dogs,1,2    ;dogs can't walk through dogs
Collisions coll_dogs,coll_trees,1,2   ;dogs can't walk through trees
Collisions coll_dogs,coll_player,1,2    ;dogs can't walk through you
Collisions coll_player,coll_trees,1,2   ;you can't walk through trees
Collisions coll_player,coll_dogs,1,2   ;you can't walk through dogs
```

As the comments explain **Collisions** sets up the object and target collision detection method and response. **Collisions coll_player,coll_trees,1,2** makes it so that the player kind can't walk through the tree kind. "1" causes ellipsoid to ellipsoid collision detection method (fastest) and the "2" is the "sliding" response style to the collision (1=STOP, 2=SLIDE, 3=Prevent from sliding down slopes).

```
EntityType cam1,coll_player
EntityRadius cam1,5
. . .
AmbientLight 50,50,50 ;up the light a tad
. . .
```

Here we set the camera player collision type and assign us a radius of 5 units. We also need to turn up the ambient world light a bit so that we can still see our special lighting effects from the previous lesson but can now more clearly observe the collision interactions.

```
Function LoadTrees(file$)
. . .
   EntityRadius tree_copy, 10
   EntityType tree_copy,coll_trees
. . .
End Function
Function LoadDogs(file$)
. . .
   EntityRadius Pack\ID, 5
   EntityType Pack\ID,coll_dogs
. . .
End Function
```

Here again in each of the Tree and Dog Load functions we set the tree and dog collision types and assign appropriate radii for each.

```
;remove previous distance limit now that dogs have collisions
Function MoveDogs(obj)
   For moveall.dog = Each dog
     ;If EntityDistance(moveall\ID,obj)>10
        PointEntity moveall\ID,obj
        MoveEntity moveall\ID,0,0,.5
        ex#=EntityX(moveall\ID):ez#=EntityZ(moveall\ID)
        PositionEntity moveall\ID,ex,TerrainY( land,ex,0,ez ),ez
     ;EndIf
   Next
End Function
```

Our final modification, now that the dogs have their own collisions, is to remove the **EntityDistance(moveall\ID,obj)>10** code from the **MoveDogs(obj)** function. This code manually kept the dogs from running through us :) . With collisions now in place it is unnecessary.

THE BIG 3 CHALLENGE!!!

- **Show the instructor your game with ALL previously accumulated personalized modifications.**

---

7. Projectiles

Whether you want to launch plasma, magic or fruit, a projectile algorithm is what you are looking for. It might sound simple enough but when you start to break it down and think it through it is quite an involved process. This could be the toughest lesson of all 9 lessons but don't try to understand it all at once. The important parts you will understand! The rest you can tackle way down the road if you really need to. We first set collision between the projectile and the target (maybe even terrain or other meshes later); create a new projectile every time the "fire key" is pressed; update the location of each new projectile, check for collisions; switch to explosion animation if collided and finally, remove the object the projectile collided with! Phew!

game7.bb

```
Global coll_player=1, coll_dogs=2, coll_trees=3, coll_projectile=4
Global skybox, land, cam1, wade_vol#, run_vol#, wadechannel, runchannel, shoot, boom
. . .
Global projectile_sprite, explosion_sprite, doghit

Graphics3D 640,480,16,2
SetBuffer BackBuffer()
. . .
Collisions coll_projectile,coll_dogs,2,1

Type Explosion
  Field alpha#,sprite
End Type

Type Projectile
  Field sprite,time_out
End Type
. . .
AmbientLight 125,125,125

LoadSprites()
. . .

While Not KeyHit(1)
  . . .
  UpdateProjectiles()
```

```
  . . .
  UpdateWorld
  RenderWorld

  ;2D stuff here
  Flip
Wend
End

Include "gamefunctions7.bb"
Include "game7data.bb"
```

gamefunctions7.bb

```
Function LoadSprites()
  explosion_sprite=LoadSprite( "explosion.bmp" )
  HideEntity explosion_sprite

   projectile_sprite=LoadSprite( "projectile.bmp" )
  ScaleSprite projectile_sprite,3,3
  EntityRadius projectile_sprite,1.5
  EntityType projectile_sprite,coll_projectile
  HideEntity projectile_sprite
End Function

Function UpdateProjectiles()
  For p.Projectile=Each Projectile
    UpdateProjectile( p )
  Next
  For e.Explosion=Each Explosion
    UpdateExplosion( e )
  Next
End Function

Function CreateProjectile.Projectile( cam1 )
  p.Projectile=New Projectile
  p\time_out=150
  p\sprite=CopyEntity( projectile_sprite, cam1 )
  EntityParent p\sprite,0
  shootChannel = PlaySound (shoot)
  Return p
End Function

Function UpdateProjectile( p.Projectile )
  If CountCollisions( p\sprite )
    If EntityCollided( p\sprite,coll_dogs )
```

```
        For k=1 To CountCollisions( p\sprite )
          doghit=CollisionEntity( p\sprite,k )
          If GetEntityType( doghit )=coll_dogs
            Exit
          EndIf
        Next
        boomChannel = PlaySound (boom)
        CreateExplosion( p )
        FreeEntity p\sprite
        Delete p
        Return
      EndIf
    EndIf
    p\time_out=p\time_out-1
    If p\time_out=0
      FreeEntity p\sprite
      Delete p
      Return
    EndIf
    MoveEntity p\sprite,0,0,2
End Function

Function CreateExplosion.Explosion( p.Projectile )
    e.Explosion=New Explosion
    e\alpha=-90
    e\sprite=CopyEntity( explosion_sprite,p\sprite )
    EntityParent e\sprite,0
    Return e
End Function

Function UpdateExplosion( e.Explosion )
    If e\alpha<270
      ez#=Sin(e\alpha)*5+5
      ScaleSprite e\sprite,ez,ez
      e\alpha=e\alpha+15
    Else
      FreeEntity e\sprite
      Delete e
      For hit.dog = Each dog
        If hit\ID = doghit Delete hit : FreeEntity doghit
      Next
    EndIf
End Function
. . .

Function LoadSFX()
```

```
   shoot=LoadSound( "shoot.wav" )
   SoundVolume shoot, .9
   boom=LoadSound( "boom.wav" )
   SoundVolume boom, .5
. . .
End Function

Function object_key_control( obj )
   If KeyHit(57)       ;spacebar = fire
      CreateProjectile( cam1 )
   EndIf
. . .
End Function
. . .
```

NEW COMMANDS

```
Global coll_player=1, coll_dogs=2, coll_trees=3, coll_projectile=4
Global skybox, land, cam1, wade_vol#, run_vol#, wadechannel, runchannel, shoot,
boom
. . .
Global projectile_sprite, explosion_sprite, doghit

Graphics3D 640,480,16,2
SetBuffer BackBuffer()
. . .
Collisions coll_projectile,coll_dogs,2,1

Type Explosion
   Field alpha#,sprite
End Type

Type Projectile
   Field sprite,time_out
End Type
. . .
AmbientLight 125,125,125

LoadSprites()
. . .

While Not KeyHit(1)
   . . .
   UpdateProjectiles()
   . . .
```

```
  UpdateWorld
  RenderWorld

  ;2D stuff here
  Flip
Wend
End

Include "gamefunctions7.bb"
Include "game7data.bb"
```

We globalize the projectile collision type (**coll_projectile=4**), sounds (**shoot, boom**), sprites (**projectile_sprite, explosion_sprite**) and the dog\ID (**doghit**) so we can remove it from our PACK when it gets hit. We create **TYPES** for our **Projectile** and **Explosion** ; with the variables for **Projectile** being the **sprite** handle itself and a **time_out** variable (so that the projectile doesn't keep going forever if it misses the target); and the variables for **Explosion** being transparency (**alpha**) and the **sprite** itself. We raise up the ambient light to see our dogs a bit better, **LoadSprites()** and then add the **UpdateProjectiles()** to the main game loop.

```
Function LoadSprites()
  explosion_sprite=LoadSprite( "explosion.bmp" )
  HideEntity explosion_sprite

   projectile_sprite=LoadSprite( "projectile.bmp" )
  ScaleSprite projectile_sprite,3,3
  EntityRadius projectile_sprite,1.5
  EntityType projectile_sprite,coll_projectile
  HideEntity projectile_sprite
End Function
```

LoadSprites() basically loads **"explosion.bmp"** into the handle **explosion_sprite** , **"projectile.bmp"** into the handle **projectile_sprite** and then hides them for when we need them later. The **projectile_sprite** is scaled, given a collision radius and collision type set.

```
Function UpdateProjectiles()
  For p.Projectile=Each Projectile
    UpdateProjectile( p )
  Next
  For e.Explosion=Each Explosion
    UpdateExplosion( e )
  Next
End Function
```

UpdateProjectiles() is really just a function that calls two other functions ever game loop cycle. It updates the position of all 'bullets" that were created by hitting the firing key and then updates each explosion that was originally a projectile before it collided with the target.

```
Function CreateProjectile.Projectile( cam1 )
  p.Projectile=New Projectile
  p\time_out=150
  p\sprite=CopyEntity( projectile_sprite, cam1 )
  EntityParent p\sprite,0
  shootChannel = PlaySound (shoot)
  Return p
End Function
```

CreateProjectile.Projectile(cam1) creates a new instance of a projectile **"projectile.bmp"** that originates from you (the camera) pointing in the same direction as **cam1**. The **time_out** countdown variable is set to a duration of 150 gameloop frames and the shooting sound is started.

```
Function UpdateProjectile( p.Projectile )
  If CountCollisions( p\sprite )
    If EntityCollided( p\sprite,coll_dogs )
      For k=1 To CountCollisions( p\sprite )
        doghit=CollisionEntity( p\sprite,k )
        If GetEntityType( doghit )=coll_dogs
          Exit
        EndIf
      Next
      boomChannel = PlaySound (boom)
      CreateExplosion( p )
      FreeEntity p\sprite
      Delete p
      Return
    EndIf
  EndIf
  p\time_out=p\time_out-1
  If p\time_out=0
    FreeEntity p\sprite
    Delete p
    Return
  EndIf
  MoveEntity p\sprite,0,0,2
End Function
```

UpdateProjectile(p.Projectile) checks for **any** collisions with **any** of the projectile sprites using **CountCollisions(p\sprite)** and **EntityCollided(p\sprite,coll_dogs)** . If the collision is a dog **If GetEntityType(doghit)=coll_dogs** then it EXITS the checking loop to **play the explosion sound**, **create the explosion** and **free up the projectile sprite**. If there is no collision the projectile is just simply moved. If the timer has run out the projectile sprite is erased.

```
Function CreateExplosion.Explosion( p.Projectile )
  e.Explosion=New Explosion
  e\alpha=-90
  e\sprite=CopyEntity( explosion_sprite,p\sprite )
  EntityParent e\sprite,0
  Return e
End Function
```

If the projectile collides with a dog, the **CreateExplosion.Explosion(p.Projectile)** is called creating an explosion sprite in the same place as the projectile sprite. The transparency is set and then updated in the next function.

```
Function UpdateExplosion( e.Explosion )
  If e\alpha<270
    ez#=Sin(e\alpha)*5+5
    ScaleSprite e\sprite,ez,ez
    e\alpha=e\alpha+15
  Else
    FreeEntity e\sprite
    Delete e
    For hit.dog = Each dog
      If hit\ID = doghit Delete hit : FreeEntity doghit
    Next
  EndIf
End Function
```

UpdateExplosion(e.Explosion) uses a special math function **Sin()** to to change the size and transparency of the explosion sprite in a very natural cyclic way. Once the transparency has cycled to a decided value then the explosion is removed, followed by the dog that was "hit".

```
Function LoadSFX()
  shoot=LoadSound( "shoot.wav" )
  SoundVolume shoot, .9
  boom=LoadSound( "boom.wav" )
  SoundVolume boom, .5
. . .
End Function
```

Simply loads and sets the volume of the shooting and explosion sounds.

```
Function object_key_control( obj )
  If KeyHit(57)       ;spacebar = fire
    CreateProjectile( cam1 )
  EndIf
. . .
End Function
```

Lastly, in the main game loop we need to pick our shooting action key (scancode 57 is spacebar) that will start the whole projectile creation process.

THE CHALLENGE!!!

- **Modify the projectile and explosion graphics and sounds to personalize this lesson.**
- **Not required, but how would you modify how big the explosion becomes or how long it lasts?**
- **Not required, but how might you get the projectiles to explode against the ground or the trees?**

---

8. Mouse Look

To add a little more professional appeal to our first person perspective, we will employ mouse pointing/aiming for the camera. This effect is common in many first person games. To complete the effect we will modify the key control so that left/right arrow now controls "strafing". To fill out the lesson we will also modify the dog movement so that it is a little more natural and we can occasionally view the sides of our pursuing dogs :)

game8.bb

```
. . .
Global projectile_sprite, explosion_sprite, doghit, aim_sprite
Global cam_x#,cam_z#,cam_pitch#,cam_yaw#
Global dest_cam_x#,dest_cam_z#,dest_cam_pitch#,dest_cam_yaw#
Graphics3D 640,480,32,1
SetBuffer BackBuffer()
. . .
LoadSprites()
. . .

While Not KeyHit(1)
  MouseTurn()
  . . .

  UpdateWorld
  RenderWorld

  ;2D stuff here
  DrawImage aim_sprite,320,240 ;the begining of a HUD ;)
  Flip
Wend
End
Include "gamefunctions8.bb"
Include "game8data.bb"
```

gamefunctions8.bb

```
Function MouseTurn()
  mxs=MouseXSpeed():mys=MouseYSpeed()
  mouse_shake=Abs(((mxs+mys)/2)/1000.0)
  dest_cam_yaw=dest_cam_yaw-mxs
  dest_cam_pitch=dest_cam_pitch+mys
  cam_yaw=cam_yaw+((dest_cam_yaw-cam_yaw)/5)
  cam_pitch=cam_pitch+((dest_cam_pitch-cam_pitch)/5)
  RotateEntity cam1,cam_pitch,cam_yaw,0

  MoveMouse 320,240

  cam_z=cam_z+((dest_cam_z-cam_z)/5)
  cam_x=cam_x+((dest_cam_x-cam_x)/5)
  MoveEntity cam1,cam_x,0,cam_z
  dest_cam_x=0 : dest_cam_z=0
  If MouseHit(1) Then CreateProjectile( cam1 )
End Function
Function LoadSprites()
  aim_sprite=LoadImage( "aim.bmp" )
  MidHandle aim_sprite
  . . .
End Function

. . .

Function MoveDogs(obj)
  . . .
  If Rand(1,100)=100 Then
    PointEntity moveall\ID,obj
  Else TurnEntity moveall\ID,0,Rand(-2,2),0
  EndIf
  . . .
End Function

. . .

Function object_key_control( obj )
  . . .
  If KeyDown( 203 )=True Then MoveEntity obj,-2,0,0    ;modified
  If KeyDown( 205 )=True Then MoveEntity obj,2,0,0
  . . .
End Function
```

NEW COMMANDS

```
Global projectile_sprite, explosion_sprite, doghit, aim_sprite
Global cam_x#,cam_z#,cam_pitch#,cam_yaw#
Global dest_cam_x#,dest_cam_z#,dest_cam_pitch#,dest_cam_yaw#
. . .
LoadSprites()
. . .
While Not KeyHit(1)
  MouseTurn()

  ;2D stuff here
  DrawImage aim_sprite,320,240 ;the begining of a HUD ;)
  Flip
Wend
End
. . .
```

As is the pattern we globalize the necessary camera and "aim" sprite variables so that the various functions can share access. **LoadSprites()** is modified to add the loading of the "aim" sprite. **MouseTurn()** will contain our new mouse aiming code that orients the camera and checks for left-button firing. **DrawImage aim_sprite,320,240** keeps the targeting "aim" sprite in the center of the screen.

```
Function MouseTurn()
  mxs=MouseXSpeed():mys=MouseYSpeed()
  mouse_shake=Abs(((mxs+mys)/2)/1000.0)
  dest_cam_yaw=dest_cam_yaw-mxs
  dest_cam_pitch=dest_cam_pitch+mys
  cam_yaw=cam_yaw+((dest_cam_yaw-cam_yaw)/5)
  cam_pitch=cam_pitch+((dest_cam_pitch-cam_pitch)/5)
  RotateEntity cam1,cam_pitch,cam_yaw,0

  MoveMouse 320,240

  cam_z=cam_z+((dest_cam_z-cam_z)/5)
  cam_x=cam_x+((dest_cam_x-cam_x)/5)
  MoveEntity cam1,cam_x,0,cam_z
  dest_cam_x=0 : dest_cam_z=0

  If MouseHit(1) Then CreateProjectile( cam1 )
End Function
```

The **MouseTurn()** aiming function looks a little complex, but makes for a smooth professional effect by **separating the moving and rotating actions of the camera**.

The Blitz functions **mxs=MouseXSpeed() : mys=MouseYSpeed()** get the **x and y speed and direction of the mouse**. The code **before MoveMouse 320,240** gets the mouse values and performs some **"smoothing"** routines and **saves the future camera values** before orienting the **rotation** of the camera. **MoveMouse 320,240 resets the "aiming" icon to the center of the screen**. The code **after MoveMouse 320,240** uses the previously generated "future" values to adjust the line **movement** (not rotation) of the camera. Also in this function we test for the left mouse button being clicked to fire a projectile **CreateProjectile(cam1)**.

```
Function LoadSprites()
  aim_sprite=LoadImage( "aim.bmp" )
  MidHandle aim_sprite
  . . .
End Function
```

We add the **"aim.bmp"** sprite to our **LoadSprites()** function list. This is the image for our targeting sprite. The black background is transparent by default. **MidHandle aim_sprite** re-centers the hotspot of **"aim.bmp"** to its actual middle rather than the upper-left corner.

```
Function MoveDogs(obj)
  . . .
  If Rand(1,100)=100 Then
    PointEntity moveall\ID,obj
  Else TurnEntity moveall\ID 0, Rand(-2,2), 0
  EndIf
  . . .
End Function
```

Here is another flourish to spice up the "natural" look of our dogs. Instead of always pointing toward you, only one in 100 loops will the dogs point toward you and the rest of the time they will randomly deviate left or right. This allows you to actually see the dogs from different angles!

```
Function object_key_control( obj )
  . . .
  If KeyDown( 203 )=True Then MoveEntity obj,-2,0,0    ;modified
  If KeyDown( 205 )=True Then MoveEntity obj,2,0,0
  . . .
End Function
```

Lastly, we go back in our **old object_key_control** function and modify it so that our left/right arrow keys now cause our camera to strafe, letting our new mouse aiming routine control the camera turning!

THE CHALLENGE!!!

- **Create your own unique AIMING icon!**

9. HUD

HUD (Heads Up Display)

The HUD is basically the 2D info overlaid on top of the 3D world, e.g. Life, Score, Ammo, Mini-Map. Most HUD updating occurs in the game loop AFTER the 3D section ending with *UpdateWorld* and *RenderWorld* commands. Creating variables to keep track of ammo, enemy count, health, inventory, etc. is essential for an active meaningful display. The graphic on the left is included in the game and will be used to keep track of your ammo, enemies and accuracy %. The black background will be transparent in the game. This lesson concludes the Blitz3D Crash Course series. Please investigate the FAQ, included help and samples to add more capability to your game!

game9.bb

```
. . .
Global projectile_sprite, explosion_sprite, doghit, aim_sprite, hud_sprite
. . .
Global Ammo#=100, DogCount#=50
. . .
fntArial=LoadFont("Arial",24)
SetFont fntArial

While Not KeyHit(1)
  . . .
  UpdateWorld
  RenderWorld

  UpdateHUD()    ;2D stuff here
  Flip
Wend
End
. . .
```

gamefunctions9.bb

```
Function UpdateHUD()
  DrawImage aim_sprite,320,240
  DrawImage hud_sprite,20,20
  Color 0,0,255
  Rect 50,30,Ammo,10
  Text 70,60,Int(DogCount)
  If Ammo<100 Then
    Text 135,60,Int(((50-DogCount)/(100-Ammo))*100)+"%"
  If DogCount=0 Then
    Color 255,255,0: Text 320,200,"You Win!!!",1,1
  If DogCount>0 And Ammo=0 Then
    Color 255,255,255: Text 320,200,"You LOSE!!!",1,1
End Function
. . .

Function LoadSprites()
  hud_sprite=LoadImage( "hud.bmp" )
. . .
End Function

. . .

Function CreateProjectile.Projectile( cam1 )
  If Ammo>0 Then
    Ammo=Ammo-1
    p.Projectile=New Projectile
    p\time_out=150
    p\sprite=CopyEntity( projectile_sprite, cam1 )
    EntityParent p\sprite,0
    shootChannel = PlaySound (shoot)
    Return p
  EndIf
End Function

. . .

Function UpdateExplosion( e.Explosion )
  . . .
  For hit.dog = Each dog
    If hit\ID = doghit Then
      Delete hit : FreeEntity doghit : DogCount=DogCount-1
  Next
  . . .
End Function
. . .
```

NEW COMMANDS

```
Global projectile_sprite, explosion_sprite, doghit, aim_sprite, hud_sprite
. . .
Global Ammo#=100, DogCount#=50
. . .
fntArial=LoadFont("Arial",24)
SetFont fntArial
. . .
  UpdateHUD()    ;2D stuff here
```

First we globalize **hud_sprite** and **Ammo#=100, DogCount#=50** to initiate a starting numerical count of the things we care about. The commands **fntArial=LoadFont("Arial",24)** and **SetFont fntArial** prepare our text size and style. **UpdateHUD()** in the main game loop will call the function we are spotlighting in this final lesson.

```
Function UpdateHUD()
  DrawImage aim_sprite,320,240
  DrawImage hud_sprite,20,20
  Color 0,0,255
  Rect 50,30,Ammo,10
  Text 70,60,Int(DogCount)
  If Ammo<100 Then
    Text 135,60,Int(((50-DogCount)/(100-Ammo))*100)+"%"
  If DogCount=0 Then
    Color 255,255,0: Text 320,200,"You Win!!!",1,1
  If DogCount>0 And Ammo=0 Then
    Color 255,255,255: Text 320,200,"You LOSE!!!",1,1
End Function
```

We are now drawing our aim and hud sprites within this function. **Rect 50,30,Ammo,10** draws a graphical ammo amount bar RECTangle on top of th hud_sprite at screen location 50,30 with a changing width (**Ammo**) and a constant height of **10**. **Text 70,60,Int(DogCount)** displays our quantity of remaining dogs. **If Ammo<100 Then Text 135,60,Int(((50-DogCount)/(100-Ammo))*100)+"%"** is a little bit of math that calculates and displays your % shooting accuracy. **If DogCount=0 Then Color 255,255,0: Text 320,200,"You Win!!!",1,1** will notify that you won if all dogs are removed. **If DogCount>0 And Ammo=0 Then Color 255,255,255: Text 320,200,"You LOSE!!!",1,1** tells you "You Lose!!!" if you run out of ammo with dogs remaining. the **1,1** "switches" at the end of each **TEXT** command allows for centering on both the X and Y axis.

```
Function LoadSprites()
  hud_sprite=LoadImage( "hud.bmp" )
. . .
End Function
```

Adds **"hud.bmp"** to our old LoadSprites() routine.

```
Function CreateProjectile.Projectile( cam1 )
  If Ammo>0 Then
    Ammo=Ammo-1
    p.Projectile=New Projectile
    p\time_out=150
    p\sprite=CopyEntity( projectile_sprite, cam1 )
    EntityParent p\sprite,0
    shootChannel = PlaySound (shoot)
    Return p
  EndIf
End Function
```

Reduce ammo by 1 each time a projectile is created by keyboard or mouse triggering.

```
Function UpdateExplosion( e.Explosion )
  . . .
  For hit.dog = Each dog
    If hit\ID = doghit Then
      Delete hit : FreeEntity doghit : DogCount=DogCount-1
  Next
  . . .
End Function
```

Finally a simple modification to our previous dog removal routine to keep track of total dogs remaining.

THE FINAL CHALLENGE!!!

- **Combine your knowledge of HUD operation creating a new layout and apply everything you have learned so far to make your own unique game!**
- **FOR FURTHER STUDY e.g. save/load, multiple levels, high scores, multiplayer....**
