AUGUST 2009 £FREE

BlitzMAX coder

0.1

Exclusive with Mark Sibly
Return of the Jedi
Bullet Candy 2
BCF 0.5
Sexy Curves

BlitzMax coder - Issue 01

BlitzMAX coder

0.1

Hi and Welcome to the first issue of the BlitzMax Coder Magazine!
In this issue we will introduce you to the world of BlitzMax development, enjoy!
blitzmaxcoder.com

News

The blitzmaxcoder.com website should be following on shortly, the aim is to provide a quality resource for BlitzMax coders, where we hope to be able to bring together well organised articles, resources, features and tutorials for you.

Hopefully I'll have more interesting news items here in the future if you have a newsworthy item you would like to see here please contact news@blitzmaxcoder.com,

If you would like to comment on any of the articles here you can contact us at feedback@blitzmaxcoder.com.

About BlitzMax

BlitzMax is a Basic programming language with Object-Oriented features and easy to use commands for graphics and sound.

Combined with an active and helpful community forum with additional modules for more advanced features.

BlitzMax is cross platform, allowing you to develop games and applications for Mac, Windows and Linux based systems.

BlitzMax can be found at: www.blitzbasic.com

Try the free demo version to see it in action.

Interview

We have managed to get a short interview with the creator of BlitzMax Mark Sibly...

Allan: BlitzMax basic is quite a complex language with OO features, threading, modules, reflection when it comes to creating it did you start out with a clear idea of what you wanted and what brought about this feature set?

Mark: When I started out, I really just wanted to simplify down what I used and/or liked about C++. This is sort of what all the Blitz's have been about - Amiga Blitz was based on my knowledge of assembly code, Blitz2D on C etc.

But BlitzMax has since evolved substantially (more than any previous version of Blitz) hopefully in interesting and useful directions! Some of that has been guided by the community (threads), some by my own ideas (reflection).

Allan: What is your favourite feature of the BlitzMax language, and why?

Mark: Garbage collection probably, because it (generally) simplifies development a whole lot.

It's taken a while to 'get right' (and is, in fact, still improving) but I believe it's a hugely valuable feature once you get your head around it.

Allan: What is your least favourite feature of BlitzMax, and why?

Mark: The 'Extern Type' hack to support COM - yuck; possibly also pointers/low level access in general.

Allan: Are there any language features that you would really like to add to BlitzMax or would include in it's successor?

Generics/templates would be nice.

This is where you don't just have TMaps/TLists of objects, but Tlist of objects of a specific type - hypothetical usage code could look like:

```
Local list:TList<TActoractors>=New TList<Actor>
list.AddLast New TActor
Local actor:TActor=list.Last()
```

Project: BlitzMax
Developer: Mark Sibly
Website: blitzbasic.com

Sections

Programming

We have 3 development articles covering the most basic Hello World example through to more advanced features.

Interviews

An interview Mark Sibly the man behind BlitzMax and Pete Carter of BlitzMax Community Framework.

Competitions

The BlitzMax Community Framework have launched there second competition with great prizes!

In the works...

We visit two developers in the middle of creating their latest games with BlitzMax.

One making a fast paced shoot-em up and the other a retro remake that never was...

Reviews

Would you like to submit your game for review@blitzmaxcoder only games made with BlitzMax please?

Feedback

Have your say at: feedback@blitzmaxcoder?

BlitzMax Coder is brought to you by Arowx.com

Interview

Project: BlitzMax
Developer: Mark Sibly
Website: blitzbasic.com

What's the point? It can often lead to faster code, as it reduces the amount of downcasting that needs to be performed (eg: last line of example above). It's also often just plain nicer coding wise to be able to have containers etc that can only be used with objects of certain types.

I'd also like to swipe := for auto typing in variable declarations, eg:

```
Local list:=New TList<Actor
Local tmp:="Hello"
For Local actor:=EachIn list
   'blah...
Next
```

Plus, get rid of the 2 level namespace limit for modules (make modules more pythonesque); use LLVM for codegen; consider making names case sensitive (I've never quite dug the whole 'T' for typenames thing); a whole bunch of stuff really...although I believe it would all remain recognizably 'BlitzMax'. Also, this would almost certainly go into a (currently hypothetical) BlitzMax2!

Allan: Quite recently threading was added to BlitzMax, I believe this is to allow people to take advantage of multiple processors, as GPU and CPU's increase their core count do you see major changes in the IT industry in the future?

Mark: No, I don't really see any major changes in the near future. More cores/threads is good, but writing code that is both safe and effective to take advantage of all these cores is pretty hard.

The trick is to find things that can be parallelized in bulk - eg:

instead of just parallelizing 2 'different' jobs such as updating/rendering, you need to work out how to parallelize n 'same' jobs such as actor updates. Sure, the first approach will (with considerable effort) be faster than a non-threaded version, but if we want speeds to get back to doubling etc. every 18 months (and we do!), the second approach is the way to go.

This is how GPUs work and why they're just still getting faster...and I guess this will eventually spill over into physics and perhaps one day even game/actor logic so...who knows?!

Allan: There is a lot of threads about requests to extend the OS/CPU base of BlitzMax beyond it's current Windows/Mac and Linux cross platform abilities, would this be technically feasable or even financially viable?

Mark: It's certainly feasible, but not really viable for me at the moment - it'd be a shedload of work for the sake of a pretty small, specialized group of customers.

That said, I wouldn't be adverse to the idea of opening up the backend code generator (the bit that converts compiler intermediate code to either x86 or powerpc asm) for people to play with.

Someone could then (theoretically - there's some pretty hardcore C++ in there!) write their own ARM (or whatever) backend, dealing to the CPU issue. Then, you'd need to rewrite a bunch of modules to deal to the OS side of things...

But it'd realistically be a LOT of work, and I doubt it'd ever be quite 'done' - which isn't to say it wouldn't be useful, it's just not something I'm keen on getting heavily involved with myself.

Allan: The forums are a great place for beginners and even advanced users to share ideas and help each other, has the Blitz community always been like this?

Yep, pretty much. Not sure why, but Blitz just seems to attract a better class of developer!

Allan: There are a lot of Modules, Engines and Tools generated by the BlitzMax community are there any areas where you think the community could really make a difference to enhancing the feature set of BlitzMax?

Mark: Hmm...not sure really...think I'll just leave it up to the community to keep on surprising me as they constantly do!

BlitzMaxCoder would like to thank Mark for his time and on behalf of all BlizMax Coders keep up the good work!

Programming

Project: Hello World I
Level: Beginner
Author: Merx

For the first programming tutorial I thought I could start with the basics...

In BlitzMax you can generate text output to the screen using the DrawText command so the classic example to display Hello World is:

```
' Hello World - Tutorial 01
Graphics 800,600

Repeat
        Cls
        DrawText("Hello World from BlitzMaxCoder.com", 250, 250)
        Flip
Until AppTerminate() Or KeyHit(KEY_ESCAPE)
```

This simple example will show a window with the desired text in about the middle of the screen.

An overview of the commands used:

```
Graphics        - creates a 2D screen with the desired width and height.
Repeat..Until   - provide a simple display loop
Cls             - clears the screens back-buffer to the default cls colour (Black)
DrawText        - Draws the text on the back-buffer using the default draw colour (White)
AppTerminate    - returns true when the user closes the window, allowing the loop to end.
KeyHit          - returns true when the user hits the ESCAPE key, allowing the loop to end.
```

A few things you might want to have a play with...

1. Change the draw colour using the SetColor or the background colour with SetClsColor.
2. Load a different font or size of font using LoadImageFont
3. Spin the text with SetRotation.
4. Change the text to a graphic image with DrawImage.
5. Change how the text is drawn with SetBlend

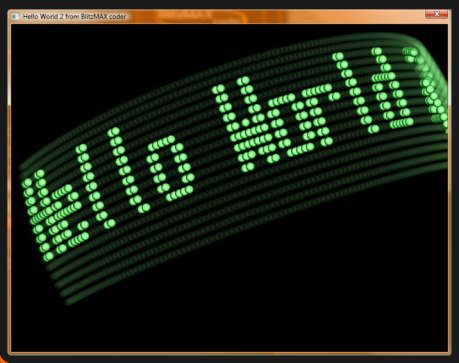
Have a play with the other drawing command and most importantly have fun.

Programming

So lets have a play and see if we can get Hello World to appear in a bit more of a retro fashion, courtesy of some nice code developed and released in the BlitzMax forums by Jesse under Graphics called 'Sphere Text'.

Here is my slightly adapted version...

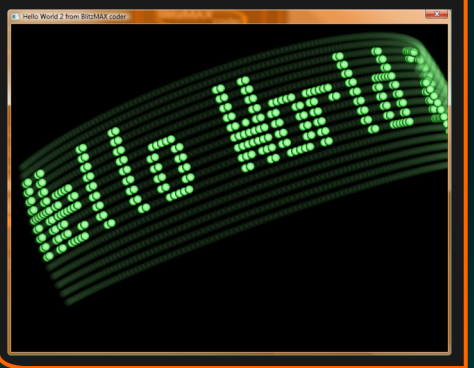
Project: Hello World 2
Level: Beginner
Author: Merx



```
' Hello World - Tutorial 02

SuperStrict

Global twidth%,Theight%
Type Tnode
	Field x!
	Field y!
	Field z!
End Type

Graphics 800,600
SetBlend ALPHABLEND

Global ledfont:TImageFont = LoadImageFont("C:\Windows\Fonts\consola.ttf", 20)

Global greenLED:TImage = LoadImage("greenLED.png")

Global format:tnode[][] = getformat()
Global textbox%[,]=gettextbox()
Local st$ = "Hello World 2... This is so amazingly cool. Found it in the code archives and just added some alpha and some LED images and wow!"
st :+ " I hope you all enjoy it."
Local n% = TextWidth(st)
Local ang% = 0

Repeat
	For Local q% = 0 To st.length-1
		put((st[q]),ang-8*q,GraphicsWidth()/2,GraphicsHeight()/2)
	Next
	Flip -1
	Cls
	ang = (ang+2) Mod (180+n)
Until KeyDown(key_escape) Or AppTerminate()

Function put(c%,ang%,offsetx%,offsety#)
	c = (c-32)*8
	ang = 180 -ang
	If ang >188 Return
	For Local x% =  0 To 7
		For Local y% = 0 To 15
			If (x+ang) < 180 And (x+ang)>0
				If textbox[x+c,y] = 0
					SetAlpha 0.1
					DrawImage greenLED,format[x+ang][15-y].x+offsetx,format[x+ang][15-y].y+offsety
				Else
					SetAlpha 1.0
					DrawImage greenLED,format[x+ang][15-y].x+offsetx,format[x+ang][15-y].y+offsety

				EndIf
			EndIf
		Next
	Next
End Function

Function getformat:tnode[][]()
	Local nodeList:TList=CreateList()
	Local d! = 1500
	Local mz! = -90
	Local AngleAxisz! = 20 'spin along y axis; angle of z
	Local AngleAxisx! = 20 'spin along z axiz; angle of x
	Local AngleAxisy! = 20 'spin along x axis; angle of y
	Local gtext:tnode[][]
	gtext = gtext[..180]
	For Local c% = 0 To 179
		gtext[c]=gtext[c][..16]
	Next
	Local sinAngleAxisx! = Sin(AngleAxisx)
	Local cosAngleAxisx! = Cos(AngleAxisx)
	Local sinAngleAxisy! = Sin(AngleAxisy)
	Local cosAngleAxisy! = Cos(AngleAxisy)
	Local node:tnode
	For Local i! = -8 To 7
		node= New tnode
		nodelist.addlast(node)
		node.x = (Cos(i*2)*25)
		node.y = (Sin(i*2)*25)
		node.z = 0
	Next

	For angleaxisz = -179 To 0 Step 1
		Local sinAngleAxisz! = Sin(AngleAxisz)
		Local cosAngleAxisz! = Cos(AngleAxisz)
		Local n% = 0
		For node  = EachIn nodeList
			Local X! = -node.x;
			Local Xa! = cosAngleAxisz * X - sinAngleAxisz * node.z;
			Local Za! = sinAngleAxisz * X + cosAngleAxisz * node.z
			Local Ya! = cosAngleAxisx * node.y - sinAngleAxisx * Xa;
			X = cosAngleAxisx * Xa + sinAngleAxisx * node.y;
			Local Z! = cosAngleAxisy * Za - sinAngleAxisy * Ya
			Local Y! = sinAngleAxisy * Za + cosAngleAxisy * Ya;
			Z = Z + mZ
			Local sx! = D * X / Z
			Local sy! = D * Y / Z
			gtext[179+angleaxisz][n] = New tnode
			gtext[179+angleaxisz][n].x = sx
```

Programming

Project: Hello World 2
Level: Beginner
Author: Merx



```
                        gtext[179+angleaxisz][n].y = sy
                        n = n + 1
                Next
        Next
        Return gtext
End Function

Function gettextbox%[,]()
                Local text$
                For Local c% = 32 To 127
                        text :+ Chr(c)
                Next
                twidth% = TextWidth(text$)
                theight% = TextHeight(text$)
                Local array%[twidth,Theight]
                Local image:TImage=CreateImage(Twidth,theight,1,DYNAMICIMAGE)
                Cls
                DrawText text,0,0
                GrabImage image,0,0
                Local map:TPixmap = LockImage(image)
                Cls
                Local c%=0
                For Local y# = 0 To PixmapHeight(map)-1
                        For Local x#=0 To PixmapWidth(map)-1
                                Local color% = $fff & ReadPixel(map,x,y)
                                If  color array[x,y]=1 Else array[x,y] = 0
                        Next
                Next
                Return array
End Function
```

This is a good example of the helpful and fun code that you can find hidden away in the copright free blitzmax code archives...

Note that for best effect you will need a greenLED.png image like this one:

Have a play with this code and see what it can do.

Don't forget to check out the other fun and helpful code samples in the code archives: http://www.blitzbasic.com/codearcs/codearcs.php

Submit Article...

Project: Your Project
Level: Any
Author: You

Calling out to all BlitzMax Coders...

Can you fill this space with a stimulating Tutorial or 'How To' on a programming, development or design topic?

Kind Regards

Allan (aka Merx)

submissions@blitzmaxcoder.com

Programming

Project: Calculating and Drawing Bezier Curves with BlitzMax
Level: Advanced
Author: ImaginaryHuman

Introduction

In the world of computer graphics, images on the screen can be created by the computer in a series of drawing commands. BlitzMax is no foreigner to graphics rendering, able to draw pixels, lines, rectangles, ovals and images. Using Plot, DrawLine, DrawRect and DrawOval BlitzMax can render filled shapes to the screen, using DrawImage to draw textured rectangles.

You can even combine these commands with transformations, including SetOrigin, SetHandle, SetScale and SetRotation. This means you can draw rotated, scaled and re-positioned lines, rectangles, ovals and images with ease. BlitzMax is even able to use the graphics hardware to draw these objects via the OpenGL or DirectX interfaces, proving to be capable at fast hardware-based rendering of multiple objects in a variety of blend modes.

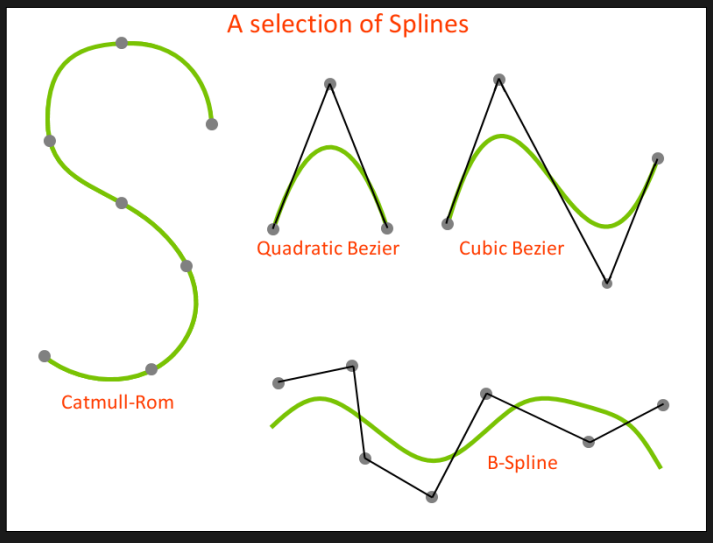
While these graphics commands may be sufficient for a variety of presentations, BlitzMax does not by default provide commands for the drawing of curved lines and objects. In addition to the desirable ability to draw curves is also the ability to calculate them, and to use the results of these calculations for other purposes.

Curves can be used to model the animation paths which objects will follow, to smoothly transition from one set of data to another, to smooth out fluctuations in performance, or even to smoothly morph one person's face into another. They have many uses, both in games and applications, and in this article we'll look at how we can create curves with BlitzMax.

Types of Splines

A spline is basically another name for a curved line. Typically the curve is designed by specifying some coordinates, called `control points`. Usually some mathematical equation then generates the curve based on the position of the control points. The curve usually begins near to the first control point, ending near the last. Adjusting the curve usually means moving the control points, which then automatically updates the curve.

The shape of the curve is influenced by the position of one or more control points along the way. Sometimes the curve will pass through all of the control points, or sometimes only the first and last, or none, depending on the type of spline. Below is a simple example of what some different types of splines may look like, accompanied by control points:



There are many different types of splines. Following this article, and for a more in-depth study covering most of their different qualities and characteristics, you might enjoy reading the following PDF document:

http://www.cs.umu.se/education/examina/Rapporter/461.pdf

There is also a significant volume of information online about the many different types of splines. You may enjoy browsing through the wikipedia pages, here:

http://en.wikipedia.org/wiki/Category:Splines

Or for the more hands-on among you, try these simple java applets which let you draw and adjust various types of splines:

http://www.cse.unsw.edu.au/~lambert/splines/

Here is the wikipedia page for Bezier curves which shows a nice animated version of the algorithm we are about to use (about 2/3rds of the way down the page):

http://en.wikipedia.org/wiki/Bezier_curve

Different types of splines lend themselves to different uses. For example, the Catmull-Rom spline passes smoothly through every control point. This type of spline can be great for drawing a smooth animation path where an object needs to continually flow through various points without stopping or experiencing sudden changes in direction.

With a few clicks of the mouse you can decide exactly which coordinates an object will pass through and it will be sure to visit all of those locations. Since the curve passes through every control point, clicking to draw the curve is very easy - it does most of the work for you automatically. The curve will smoothly ease into and out of each control point gently. The direction in which the curve eases into a point depends also on the direction it will need to take afterwards toward the next control point.

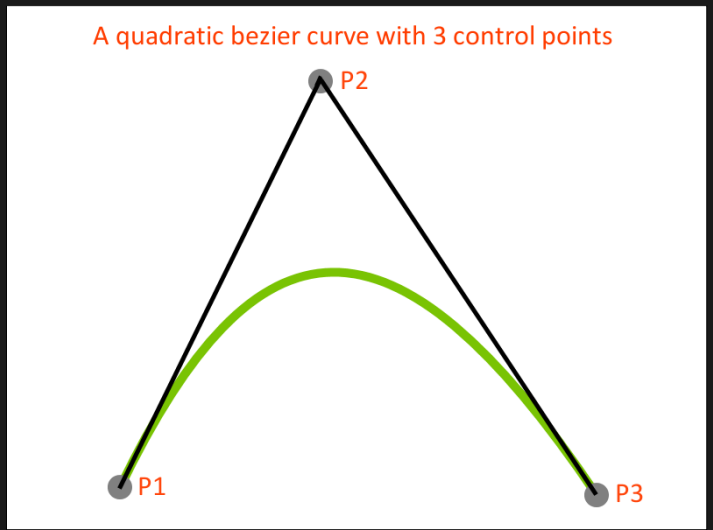
On the other hand, sometimes you might prefer that the curve does not smoothly pass through every point. You might be designing a logo where you need a sharp right-angled corner rather than a curved one, and yet continue with a curved edge afterwards. Or you might need a sharp point in amongst flowing curves. For this you might prefer to use Bezier curves with multiple individual curve sections joined together.

Sophisticated graphics applications such as used for ray-tracing, vector graphics, 3D modelling and animation often make use of Bezier curves, b-splines and NURBS (Non-Uniform Rational B-Splines). Each offers a unique way to define the curves and to manipulate their shape. Some people prefer different types of splines because of the way you can manipulate them, or because of the subtle control offered over their shape.

Although the many different types of splines each have interesting properties, covering them all is beyond this article and I recommend reading the above articles if you are interested. For our purposes, we'll take a look at the relatively simple and easy-to-grasp Bezier curves.

Bezier Curves

The Bezier curve was invented by the mathematician Pierre Bezier. A Bezier curve segment begins at a first control point, is influenced by one or more other control points (which it does not pass through), finally ending at the last control point. The number of control points in a single Bezier curve is often 3 or 4, but can be more. At it simplest, we'll take a look at a 3-point Bezier curve, otherwise known as a quadratic curve.



Here is an example of how a 3-point Bezier is put together:

Programming

Project: Calculating and Drawing Bezier Curves with BlitzMax
Level: Advanced
Author: ImaginaryHuman

P1 is the first point of the curve. P3 is the last point of the curve. P2 is an in-between point which, although the curve does not pass through it, attracts the curve to it as if magnetic. If you imagine a straight line from P1 to P3, you can then imagine that P2 is attracting that line towards itself, causing it to bend. Essentially, curves are generated by calculating the influence of each control point along the way.

The positions of the control points determines the shape of the curve. The curve is adjusted by moving the control points. You generally cannot just click on the curve to drag it into a new shape, although such higher-level functionality can be programmed. To modify the curve, you would normally click on a control point and move it to a new position - the curve adjusts and updates automatically. Curve editing is somewhat indirect with Beziers - only the first and last control points are passed through. They are suited to a little more detailed tweaking than the Catmull-Rom spline which makes them good for generating graphics.

With three points defining a curve, you generally can create an arc covering up to 180 degrees of curvature. A curve which starts out by heading in one direction can end by returning in the opposite direction - the two end-points can be at the same location. To create more complex curves you can simply create further Beziers following on from each other, sharing the end-point of the previous curve and re-using it as the starting point of a new curve.

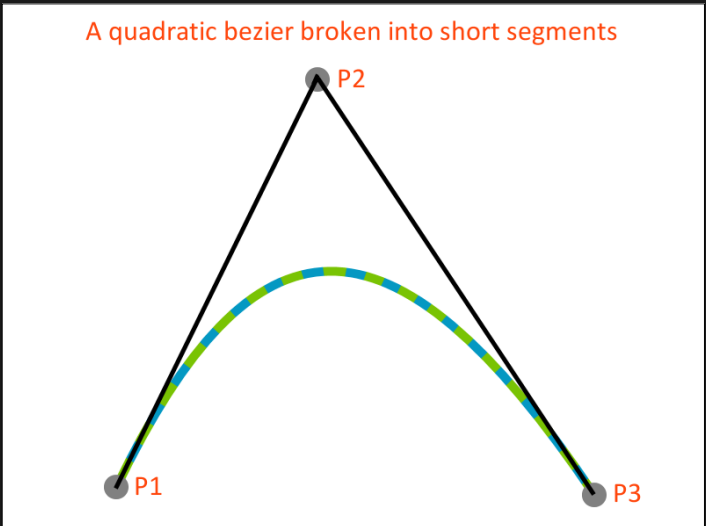
Drawing a 3-Point Bezier Curve

The drawing of a Bezier curve is not as easy as simply drawing lots of pixels. It's also not as easy as just asking the graphics hardware to draw a curve - it's generally not capable of drawing anything other than points, lines or triangles. How does a straight-edged line represent a curve? Depending on the algorithm, there is usually some math involved, and it's not always easy to understand, especially for a beginner. We won't be getting into any difficult math here!

We also need to consider that a curve is really a virtual mathematical line, with potentially infinite accuracy, and no width. Zooming into the curve really should continue to show us perfectly smooth curvature no matter how far we zoom in. And indeed that is one of the most desirable properties of curves - vector graphics can be zoomed in or out significantly and still show accurate smooth curves - not the jagged pixels which you'd see from a zoomed bitmap.

A curve is really a single continuous entity, which does not automatically map to a screen of separate digital pixels. It's kind of like the difference between analog and digital. The only way we can draw it on the screen is to approximate where the curve will be as it crosses each individual pixel at a specific resolution and size. We need to `modulate` it (sample it), like a modem would modulate an analog signal and turn it into a digital one. Drawing a curve means converting from an analog curve to a digital curve.

Basically, almost all methods of drawing a curve involve starting at the beginning point, gradually progressing along the curve by varying a value in an equation, and stopping at the end point. By just varying one value - the `position along the curve`, we can `step` along the curve, drawing as we go. This means that to draw it, we have to draw lots of little `pieces` of the curve in succession, breaking the continuous curve into little chunks, as above.



In order to calculate the curve - or rather, several fixed points along the curve, we have to make some kind of decision about what size the step will be, and thus how much distance we travel in each step along the curve. This distance will determine how we draw each little piece of the curve.

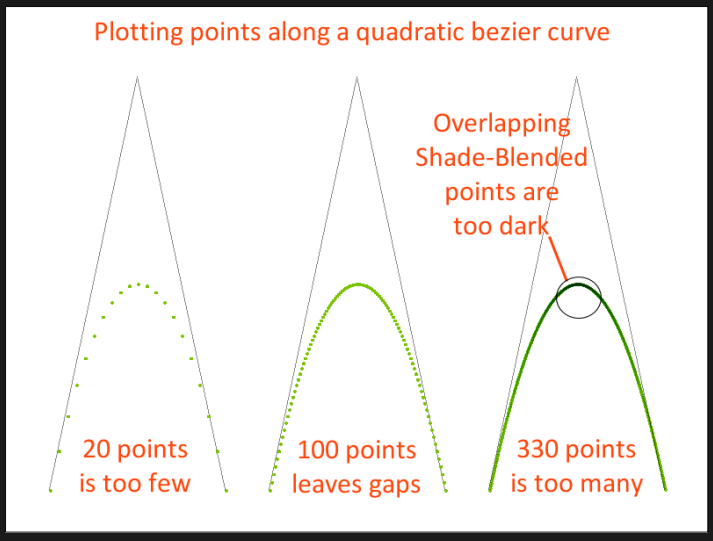
Unfortunately we can't very easily say `step 1 pixel along the curve`. It's not as easy as drawing straight lines or rectangles, where you can just step one pixel at a time. The curve has a different length which is not necessarily obvious based on the control points. Prior to actually calculating the whole curve, it is difficult or impossible to know how many pixels it will cover - a catch 22.

To know the curve length first, you'd end up having to calculate the whole curve twice, which is undesirable. The more mathematically heavy approaches to curve calculation can use a large number of additions and multiplications - hundreds or even thousands per curve. We need a way to both calculate and draw the curve as we travel along it but without knowing the length - which usually means guessing the step size.

We have to decide upon some kind of step value which we think will be an acceptable size, and this can vary depending on screen resolution, performance, the sharpness of the curve and how smooth we want the curve to look. We need a value which produces a good enough accuracy to be able to represent the curve as `smooth` visually. Obviously in most cases we don't want the curve to look jagged or angular - unless it doesn't matter in what you're using them for.

If we make our step very small and use Plot to draw the pixels, we are hoping that the step will be small enough not to produce any gaps in the curve. If we happen to pick a step which isn't short enough we might skip a few pixels along the way, breaking the curve. Then again, too small a step when it isn't necessary can mean you're drawing the same pixel over and over again. That especially doesn't work well when you're also drawing in a blend mode like LightBlend or ShadeBlend, yet alone being inefficient.

Here are some examples of how drawing individual points might produce desirable or undesirable results (I used DrawOval X,Y,3,3 for each point to make it more obvious, rendering with ShadeBlend to show overlapping points):



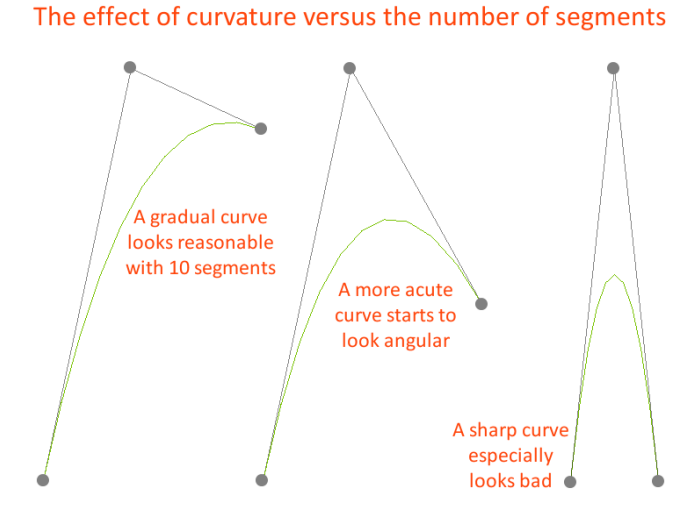
Another option is to draw the curve by linking those spaced-out Plots together - drawing lots of short straight lines one after the after. Obviously if these lines are long, or not short enough to represent a shaper curve, we're going to start noticing that we've drawn a bunch of straight edges and not perfectly curved ones - although overall it would approximately represent a curve.

With some experimentation you can arrive at a step size which produces short enough lines as to not be very noticeable for most curves. In general,

Programming

Project: Calculating and Drawing Bezier Curves with BlitzMax
Level: Advanced
Author: ImaginaryHuman

the sharper the angle of the curve the smaller the step must be in order to smoothly represent it. Gradual curves work okay with longer straight line segments. The following diagram demonstrates the effect of different curves with the same step size (10 segments per curve ? a step of 0.1):



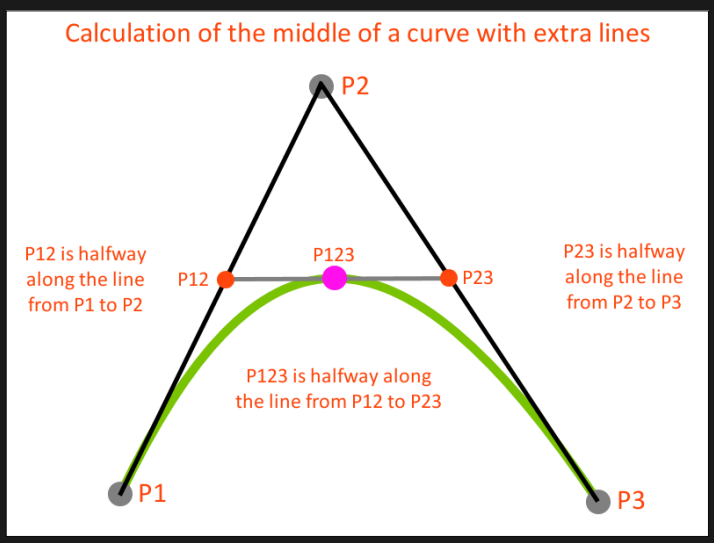
It is quite difficult visually to tell the difference between a curve made of short straight lines and one made of short curves. In fact, BlitzMax's DrawOval command doesn't really draw perfectly curved ovals, it draws lots of straight edged shapes which, together, approximately looks like an oval - and the size of each of these sub-shapes is small enough that you can't very easily tell the difference. It might not be obvious, but BlitzMax's ovals are actually made of straight-edged triangles.

We need to use a similar trick to draw our curves, converting from a perfect mathematical curve to an approximate curve made of little straight lines - something the graphics hardware can actually draw.

The Bezier Curve Algorithm

Now we are ready to look at an algorithm for calculating a Bezier curve. We are not going to use the traditional mathematical equations for Bezier curves. Instead, there is a simpler and easier-to-understand method for calculating them. So long as you know how to add, subtract and multiply floating-point numbers with BlitzMax, you should be all set.

This simpler method, and usually more efficient also, was first publicized by a guy called De Casteljau. I also arrived at this method independently, but he beat me to it. What he observed is that you can produce a Bezier curve through a simple series of subdivisions of straight lines.



To understand this, first take a look at this diagram, showing a new line and some new points, which we will need to calculate in order to arrive at a point on the curve. In this example we'll show the mid-point of the curve, found by simply dividing the length of the two sides by 2 (or multiplying by 0.5) and then finding the middle of the line between those two points (see above).

Although in this diagram we are finding the middle of the curve with a position along the curve of 0.5, we can use any other values between 0 and 1 to find the whole curve. If we wanted to find a point 25% of the way along the curve, we would first find the length of the line `P1 to P2` and then step along that line by multiplying the length by 0.25 instead of 0.5. We do the same with the line from P2 to P3. The new line which joins these points is then on a diagonal. We can find the point on the curve by multiplying the length of the new line by 0.25 instead of 0.5.

Since we are going to be drawing short lines to build our curve, it is worth realizing that when we're at the start of the curve, at point P1, we can't draw our first line yet. It isn't until we've taken a step along the curve and calculated a second coordinate that we can then draw our first line. Also the first point P1 is already calculated (it's the control point's coordinates), so you don't need to calculate it as a point on the curve.

First of all we need a step value - so for simplicity let's say we want to represent our curve with 5 short lines. For handy maths purposes we would like our step value to begin at 0 and end at 1.0. Later we can just multiply by this number to do part of the algorithm. 0 means we are at the start of the line, and 1.0 means we are at the end of the line.

So if we're going to have 5 line segments, our step will be 1.0/5.0 which is a step of 0.2. Our Bezier will actually consist of 6 points along the curve, which will be 0, 0.2, 0.4, 0.6, 0.8 and 1.0. These will be controlled by three control points, represented by 5 straight lines.

The algorithm in general has you gradually incrementing a `position on the curve` value, ranging from 0 to 1. We will be adding our step value to it each time and then will use that result to calculate the actual coordinates of a point on the curve.

For each step we take, we start out by measuring the distance between the first point P1 and the second point P2 - we'll call it P12. Since coordinates have both X and Y components, we need to find the difference between P1 and P2's X coordinates, and then also the difference between P1 and P2's Y coordinates. This gives us two distances, and we can find these with a couple of simple subtractions:

```
P12X = P2X - P1X
P12Y = P2Y - P1Y
```

We then need to find the point along each of these distances where we've stepped to so far. This is a simple matter of multiplying the two distances by our `position on the curve` value. So let's say we just added the step of 0.2 to our position (which begins at 0). Now we're at 0.2 on the curve. We multiply 0.2 by the distance between P1's X and P2's X, and then we multiply 0.2 separately by the distance between P1's Y and P2's Y, as such:

```
P12X = 0.2 * P12X
P12Y = 0.2 * P12Y
```

This basically has us stepping along the line between P1 and P2 by the same amount that we have stepped along our curve so far. In a way, once we get to 1.0 on the curve, we will have also arrived at point P2 along the line between P1 and P2.

Now we find the distance between P2 and P3 in the same way as before - just subtracting the X coordinates from each other, and then the same with the Y coordinates. Then we just need to multiply each of these distances by our `position on the curve value`. This gives us point P23, for example:

```
P23X = P3X - P2X
P23Y = P3Y - P2Y
P23X = 0.2 * P23X
P23Y = 0.2 * P23Y
```

We can now visualize an imaginary straight line between our two new points - from P12 to P23. What I can now tell you is that somewhere along this new line is a new point, P123, which is exactly on the Bezier curve, in this case it will be 20% of the way along the line. All we have to do is find

Programming

Project: Calculating and Drawing Bezier Curves with BlitzMax
Level: Advanced
Author: ImaginaryHuman

that point.

We simply do the same thing we did before with the other distances, using the new line between P12 and P23. You just find the difference between P12's X and P23's X, then the difference between P12's Y and P23's Y. This time around, however, we also have to add-in the points P1 and P2 so that our distance will be correct (because P12 and P23 are relative, we need them to be absolute coordinates). Then it's just a matter of multiplying these two distances by our `position on the curve` value, for example:

P123X = (P2X+P23X) - (P1X+P12X)
P123Y = (P2Y+P23Y) - (P1Y+P12Y)
P123X = 0.2 * P123X
P123Y = 0.2 * P123Y

This produces a final point, P123. Before we can think of it in terms of screen coordinates, we simply need to add some coordinates to it to change back from relative distances to absolute screen coordinates, like so:

P123X = P123X + P1X + P12X
P123Y = P123Y + P1Y + P12Y

We add P1 and P12 to the coordinates of the curve point. P1 is added so that the whole curve starts at P1 and not at 0. Then we add in P12 because we need to step down to the next level of subdivision and add that point as an offset. We're just adding up the steps we need to take to get to P123 by taking the shortest route from P1. ie we add P1, then P12, then finally P123. The coordinate P123X,P123Y is now the actual point on the curve in screen coordinates.

The basic broken-down BlitzMax sourcecode to calculate a single point on a 3-point quadratic Bezier curve is as follows:

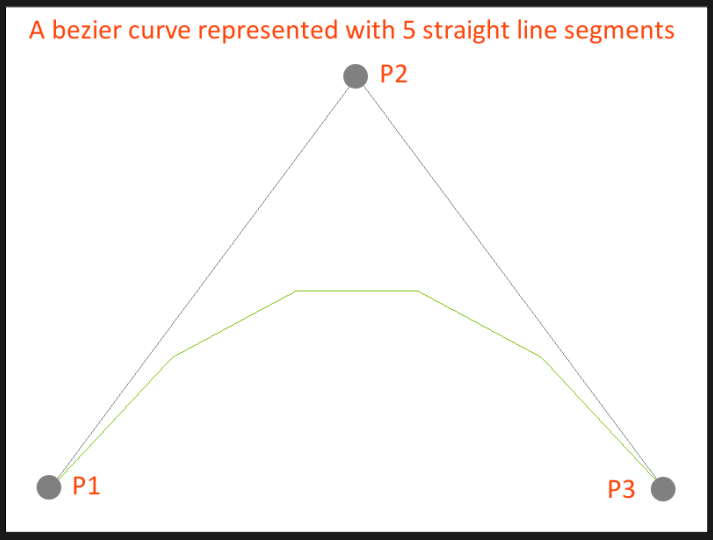
```
Function
QuadraticBezierPoint(P1X:Float,P1Y:Float,P2X:Float,P2Y:Float,P3X:Float,P3Y:Float
,PositionOnCurve:Float,P123X:Float Var,P123Y:Float Var)
  'Calculate a point on a quadratic bezier curve
  'Requires three control points with X and Y coordinates
  'and a current position on the curve in the range 0..1
  'Coordinates are returned in P123X and P123Y variables

  'Calculate distances between control points
  Local P12X:Float=P2X-P1X        'Distance in X between P1X and P2X
  Local P12Y:Float=P2Y-P1Y        'Distance in Y between P1Y and P2Y
  Local P23X:Float=P3X-P2X        'Distance in X between P2X and P3X
  Local P23Y:Float=P3Y-P2Y        'Distance in Y between P2Y and P3Y

  'Find position along the lines from P1 to P2, and from P2 to P3
  P12X:*PositionOnCurve           'Multiply position by P12 X distance
  P12Y:*PositionOnCurve           'Multiply position by P12 Y distance
  P23X:*PositionOnCurve           'Multiply position by P23 X distance
  P23Y:*PositionOnCurve           'Multiply position by P23 Y distance

  'Calculate distance between the two positions
  P123X=(P2X+P23X)-(P1X+P12X)     'Distance in X between P12X and P23X
  P123Y=(P2Y+P23Y)-(P1Y+P12Y)     'Distance in Y between P12X And P23X

  'Find position along the line from P12 and P23
  P123X:*PositionOnCurve          'Multiply position by P123 X distance
  P123Y:*PositionOnCurve          'Multiply position by P123 Y distance

  'Add up the relative positions to make screen coordinates
  P123X:+P1X+P12X
  P123Y:+P1Y+P12Y

  'The coordinate P123X,P123Y is the location of the point on the curve
End Function
```

To use this function you simply need to pass three pairs of coordinates for the control points, a position along the curve (from 0 to 1), and then a couple of variables into which the point's X and Y coordinates will be returned.

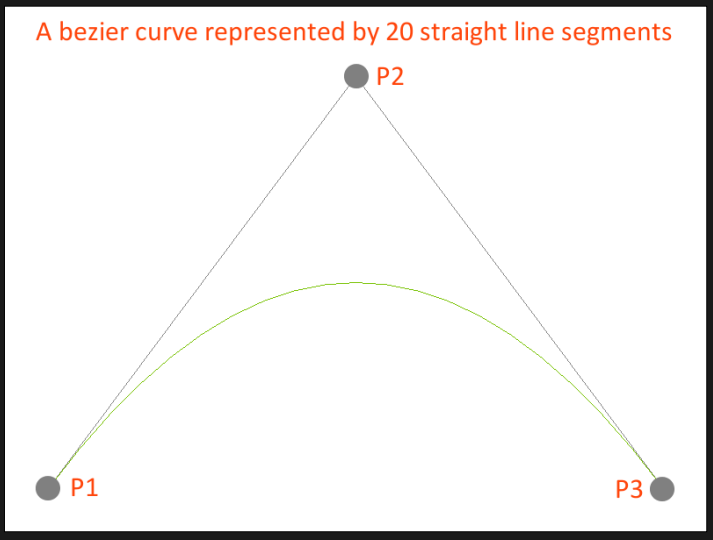
Here is a more compact version of the same code:

```
Function
QuadraticBezierPoint(P1X:Float,P1Y:Float,P2X:Float,P2Y:Float,P3X:Float,P3Y:Float
,Position:Float,P123X:Float Var,P123Y:Float Var)
  Local P12X:Float=(P2X-P1X)*Position
  Local P12Y:Float=(P2Y-P1Y)*Position
  Local P23X:Float=(P3X-P2X)*Position
  Local P23Y:Float=(P3Y-P2Y)*Position
  P123X=(((P2X+P23X)-(P1X+P12X))*Position)+P1X+P12X
  P123Y=(((P2Y+P23Y)-(P1Y+P12Y))*Position)+P1Y+P12Y
End Function
```

Now that we know how to find a point on the curve, we can actually draw a line from the previous `position on the curve` (the coordinate of point P1 in this case), to the new position on the curve - point P123. Our first piece of the curve is then complete. We now repeat this whole process all over again, continually adding the step value to the position on the curve and then multiplying this by the three sets of distances to turn it into a final coordinate. Running the loop 5 times gives us our 5 segments:



Hey presto, a Bezier curve - or something close to one. We could simply make our step size smaller to make this look really smooth - here is an example of how it would look with a step size of 0.05 (with 20 segments):



Here is an interactive program which will allow you to click on and drag the control points to adjust a quadratic Bezier curve, with 20 line segments:

```
Local NumberOfSegments:Int=20


SetGraphicsDriver GLMax2DDriver()
Graphics 800,600
SetClsColor $FF,$FF,$FF


Local PX:Float[3]
Local PY:Float[3]
PX[0]=50
PY[0]=550
PX[1]=400
PY[1]=80
PX[2]=750
PY[2]=550


Local Position:Float
Local X:Float
Local Y:Float
Local X2:Float
Local Y2:Float
```

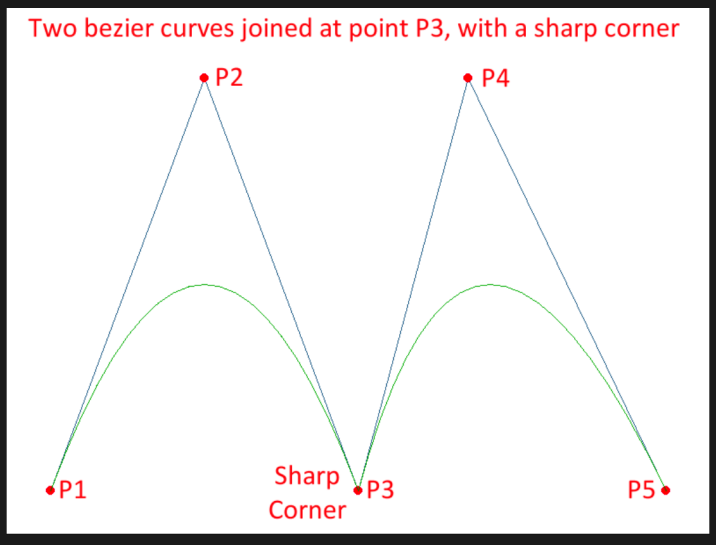
Programming

Project: Calculating and Drawing Bezier Curves with BlitzMax
Level: Advanced
Author: ImaginaryHuman

```
Local Controlling:Int=False
Local ControlPoint:Int=0
Local MX:Int
Local MY:Int
Local Point:Int
Local SegmentSize:Float=1.0/NumberOfSegments
Repeat
  'Control it
  MX=MouseX()
  MY=MouseY()
  If Controlling=False
        If MouseDown(1)
          For Point=0 To 2
         If MX>=PX[Point]-15 And MY>=PY[Point]-15 And MX<=PX[Point]+15 And
MY<=PY[Point]+15
           Controlling=True
           ControlPoint=Point
         EndIf
       Next
    Else
      Controlling=False
    EndIf
  EndIf
  If Controlling=True
    PX[ControlPoint]=MX
    PY[ControlPoint]=MY
    If Not MouseDown(1) Then Controlling=False
  EndIf


  'Draw stuff
  Cls
  SetColor $88,$88,$88
  DrawLine PX[0],PY[0],PX[1],PY[1]        'Draw lines
  DrawLine PX[1],PY[1],PX[2],PY[2]
  SetColor $FF,$0,$0
  DrawOval PX[0]-5,PY[0]-5,10,10  'Draw control points
  DrawOval PX[1]-5,PY[1]-5,10,10
  DrawOval PX[2]-5,PY[2]-5,10,10
  SetColor 121,195,3

  'Draw curve
  X=PX[0]
  Y=PY[0]
  Position=SegmentSize
  While Position<=1.00001 'Go to 1.00001 to make sure final line is drawn
    QuadraticBezierPoint(PX[0],PY[0],PX[1],PY[1],PX[2],PY[2],Position,X2,Y2)
    DrawLine X,Y,X2,Y2
    X=X2
    Y=Y2
    Position:+SegmentSize
  Wend

  Flip 1
Until KeyHit(KEY_ESCAPE) Or AppTerminate()


Function
QuadraticBezierPoint(P1X:Float,P1Y:Float,P2X:Float,P2Y:Float,P3X:Float,P3Y:Float,
Position:Float,P123X:Float Var,P123Y:Float Var)
  Local P12X:Float=(P2X-P1X)*Position
  Local P12Y:Float=(P2Y-P1Y)*Position
  Local P23X:Float=(P3X-P2X)*Position
  Local P23Y:Float=(P3Y-P2Y)*Position
  P123X=(((P2X+P23X)-(P1X+P12X))*Position)+P1X+P12X
  P123Y=(((P2Y+P23Y)-(P1Y+P12Y))*Position)+P1Y+P12Y
End Function
```

A Sequence of Beziers

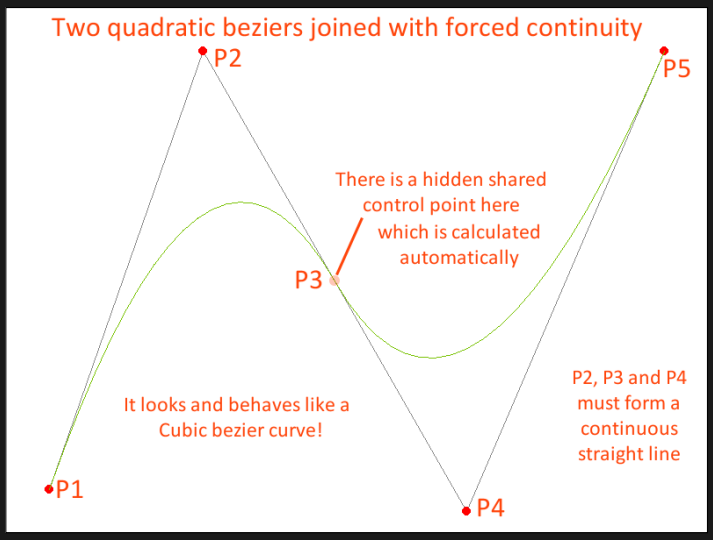
Sometimes it is desirable to have much longer smooth curves with many twists and turns. Obviously you can't do this with a single 3-point Bezier curve - it can only draw an arc and can't reverse the direction of its curvature like a cubic Bezier can.



To create more complex, smoothly-flowing curves using 3-point Beziers, we are faced with the fact that even if two curves share an end-point the curve won't necessarily flow smoothly between them, as in the diagram above (which looks like the logo for Motorola)

There is one special case where the two curves actually would flow smoothly, and that's when the line between P2 and P3 in the first curve is extended exactly into the second curve's P1-to-P2 line. So long as these two lines are at the same angle, forming a straight line between the first curve's P2 and the second curves P2, passing through the shared P3, the Bezier curve will flow naturally and `continuously`. This is an emulation of what is called `continuity` in spline lingo. While Bezier's don't naturally have continuity at their end points, we can emulate it with a bit of extra work.

A Catmull-Rom spline has this kind of smooth continuity all the time for every control point. We can emulate a Catmull-Rom spline, ie making it seem like the curve always flows smoothly through every control point, using Bezier curves. We would have to make the two lines which join to the shared control point be at the same straight angle and make the length of those two lines the same. Technically we might also have to calculate the middle control points automatically (e.g. P2) based on the angle of the two lines and where they coincide, and adjust the position of the middle control points, but in general here's how it would look:



And here is an interactive program to show how we can join two quadratic Bezier curves together using this technique - we simply force the position of the shared point to be calculated based on the distance between the two surrounding points, divided in half:

```
Local NumberOfSegments:Int=20


SetGraphicsDriver GLMax2DDriver()
Graphics 800,600
SetClsColor $FF,$FF,$FF


Local PX:Float[5]
Local PY:Float[5]
PX[0]=50
PY[0]=550
PX[1]=225
PY[1]=50
PX[2]=370
PY[2]=300
PX[3]=525
PY[3]=575
PX[4]=750
PY[4]=50


Local Position:Float
Local X:Float
```

Programming

```
Local Y:Float
Local X2:Float
Local Y2:Float
Local Controlling:Int=False
Local ControlPoint:Int=0
Local MX:Int
Local MY:Int
Local Point:Int
Local SegmentSize:Float=1.0/NumberOfSegments
Repeat
  Cls

  'Control it
  MX=MouseX()
  MY=MouseY()
  If Controlling=False
    If MouseDown(1)
      For Point=0 To 4
        If MX>=PX[Point]-15 And MY>=PY[Point]-15 And MX<=PX[Point]+15 And
MY<=PY[Point]+15 And Point<>2
          'Control the point so long as it's not the middle shared point
          Controlling=True
          ControlPoint=Point
        EndIf
      Next
    Else
      Controlling=False
    EndIf
  EndIf
  If Controlling=True
    SetColor $0,$0,$0
    DrawText MX+","+MY,0,32
    PX[ControlPoint]=MX
    PY[ControlPoint]=MY
    If Not MouseDown(1) Then Controlling=False
  EndIf


  'Force continuity - calculate the middle shared point automatically
  PX[2]=PX[1]+((PX[3]-PX[1])*0.5)
  PY[2]=PY[1]+((PY[3]-PY[1])*0.5)


  'Draw stuff
  SetColor $88,$88,$88
  DrawLine PX[0],PY[0],PX[1],PY[1]  'Draw lines
  DrawLine PX[1],PY[1],PX[2],PY[2]
  DrawLine PX[2],PY[2],PX[3],PY[3]
  DrawLine PX[3],PY[3],PX[4],PY[4]
  SetColor $FF,$0,$0
  DrawOval PX[0]-5,PY[0]-5,10,10  'Draw control points
  DrawOval PX[1]-5,PY[1]-5,10,10
  'DrawOval PX[2]-5,PY[2]-5,10,10 'Don't draw middle shared point
  DrawOval PX[3]-5,PY[3]-5,10,10
  DrawOval PX[4]-5,PY[4]-5,10,10
  SetColor 121,195,3

  'Draw curve 1
  X=PX[0]
  Y=PY[0]
  Position=SegmentSize
  While Position<=1.00001 'Go to 1.00001 to make sure final line is drawn
    QuadraticBezierPoint(PX[0],PY[0],PX[1],PY[1],PX[2],PY[2],Position,X2,Y2)
    DrawLine X,Y,X2,Y2
    X=X2
    Y=Y2
    Position:+SegmentSize
  Wend

  'Draw curve 2
  X=PX[2]
  Y=PY[2]
  Position=SegmentSize
  While Position<=1.00001 'Go to 1.00001 to make sure final line is drawn
    QuadraticBezierPoint(PX[2],PY[2],PX[3],PY[3],PX[4],PY[4],Position,X2,Y2)
    DrawLine X,Y,X2,Y2
    X=X2
    Y=Y2
    Position:+SegmentSize
  Wend


  Flip 1
Until KeyHit(KEY_ESCAPE) Or AppTerminate()


Function
QuadraticBezierPoint(P1X:Float,P1Y:Float,P2X:Float,P2Y:Float,P3X:Float,P3Y:Float,
Position:Float,P123X:Float Var,P123Y:Float Var)
  Local P12X:Float=(P2X-P1X)*Position
  Local P12Y:Float=(P2Y-P1Y)*Position
  Local P23X:Float=(P3X-P2X)*Position
  Local P23Y:Float=(P3Y-P2Y)*Position
  P123X=(((P2X+P23X)-(P1X+P12X))*Position)+P1X+P12X
  P123Y=(((P2Y+P23Y)-(P1Y+P12Y))*Position)+P1Y+P12Y
End Function
```

We could use this to create an animation path for a sprite - the sprite would be sure to always pass smoothly through every ***shared*** control point, but it would not pass through the middle control points. We could also keep adding more and more Beziers, each sharing the end point of the previous Bezier, forming a long and complicated winding curve.

Project: Calculating and Drawing Bezier Curves with BlitzMax
Level: Advanced
Author: ImaginaryHuman

When we join together quadratic Bezier curves like this, they look and act similar to how a Cubic Bezier curve would act, based on four control points - they technically might not be exactly the same shape but the editing functionality is similar. Alternatively, it is fairly trivial to add extra intermediary points and lines to the algorithm to provide support for cubic beziers. You could even use more than four points, as shown in the animations on the Wikipedia page for bezier curves.

Conclusion

There are many uses for Bezier curves and you can even create different ways to adjust them based on indirectly moving the control points. Applications like Inkscape allow you click on the curve itself and drag it to a new position. This of course requires some extra programming. You could also come up with other ways to adjust the curve suited to your needs.

It is also possible to do other fancy things with Beziers such as splitting the curve at a given point, joining curves together into one, and adding further editing features such as are found in programs like Inkscape.

Beyond this is the challenging topic of how to fill a solid object composed from Bezier curves. This requires a process of tessellation to turn the object into renderable triangles. We can even create three-dimensional curved surfaces (commonly called patches) using a grid of Beziers, which are then tessellated into a triangle mesh. These topics are big enough that we'll leave them for a future article.

Hopefully, following this article you will be able to implement Bezier curves into your future games, applications, and editors without too much trouble. Remember there are many resources online and the BlitzMax community is always ready to help.

In the works...

This is our first interview and it's with Charlie (aka jkrankie) the developer of Bullet Candy and who is currently developing the sequel Bullet Candy 2.

Check out his site www.chaliesgames.com

Allan: Could you give us a bit of background about yourself and what brought you round to using BlitzMax for game development?

Charlie: Ok then, I'm Charlie and I live in the UK. I always had a passing interest in making games and programming, but never really tried much until about 6 or 7 years ago. I Learned some Pascal at sixth form when I was a teenager, but it wasn't very interesting stuff (mostly leaning how to program databases etc.) and didn't really attempt any more programming until I was in my early 20s. I played around with C/C++, various forms of basic and got the programming bug. Shortly afterwards I left my job as a gardener to study for a Software engineering degree.

I guess I stumbled across BlitzMax around the time I started the course, and is largely responsible for me getting a 2nd rather than a 1st degree ;)

During the 2nd year of my degree, I wrote and released Bullet Candy, which was written in BlitzMax. Also more recently I've been trying to get some of the stuff that I've done since then out as freeware, the first of these releases is Space Phallus.

Allan: Bullet Candy 2 is your latest game in development, what inspired you to write it?

Charlie: I don't think I was really 'inspired' to write it, it just sort of happened. After I released the most recent builds of Space Phallus, I downloaded MiniB3d, and started playing around with that. I worked out how to mimic 2d coordinates, and plugged the old Bullet Candy code in and made a few quick 3D models of the old enemies to see what It looked like. That took just less than a week for it to be fully complete (including all the menus and achievements etc.), so I carried on playing about with it. As I learned more, I started to add more and more stuff, and it just sort of gone on from there really.

Allan: Could you give us a brief overview of the tools you use or have created to construct Bullet Candy 2?

Charlie: OK, well aside from BlitzMax and MiniB3d, I've been using a few of Brucey's Modules, namely Bass, Sqlite, and Locale. I expect I'll be adding his Volumes module once I get closer to finishing as well.

On the actual tools side, I use one that I wrote, which is essentially a stripped down version of the game, that I set to only load the media I need to work with. I use this to program enemies and levels before adding them to the main project. I also use Blide Plus, which saves me a lot of time, Wings3D for modelling, Photoshop and MS Paint for 2d stuff and a variety of
different audio tools for making sounds.

Allan: Bullet Candy 2 is a very fast paced great looking shooter, could you outline the technical challenges you have had in it's development?

Charlie: Not too many technical challenges really, MiniB3D is pretty competent for what I'm using it for. The only issue I'm having with it is that the sprite stuff isn't really fast enough for the amount of particles I want to throw around, so I've been dipping my toes into the murky waters of OpenGL trying to come up with something a bit quicker.

I tend to find the most challenging part is coming up with a nice framework that is easy to use, and won't end up irritating me during development. This usually takes a few goes to get right. Also I have a bad history with working to a design, I tend to find I work better if I'm working with a loose goal in mind. Coming up with unique levels that are fun to play is also tricky. I tend to spend more time thinking what would be fun, and work well in context than I do actually coding the levels. Other than that though, things have been pretty smooth sailing on this one :)

Allan: You mentioned that you are building up a framework and have used a number of third party module's, have you been following the Community Framework and have you consider using it or contributing to it?

Charlie: I've been following it a little (and I helped judge the first competition they ran), although I haven't used the community framework. I tend to re-use my own stuff, customising it as necessary. I've built the most generic/re-usable functions and types into a module.

Actually, looking at it just now, there is no indication on the website of what it actually contains. Perhaps they ought to put up a list of planned/existing functions?

Allan: How have you found using the BlitzMax language and forums for development?

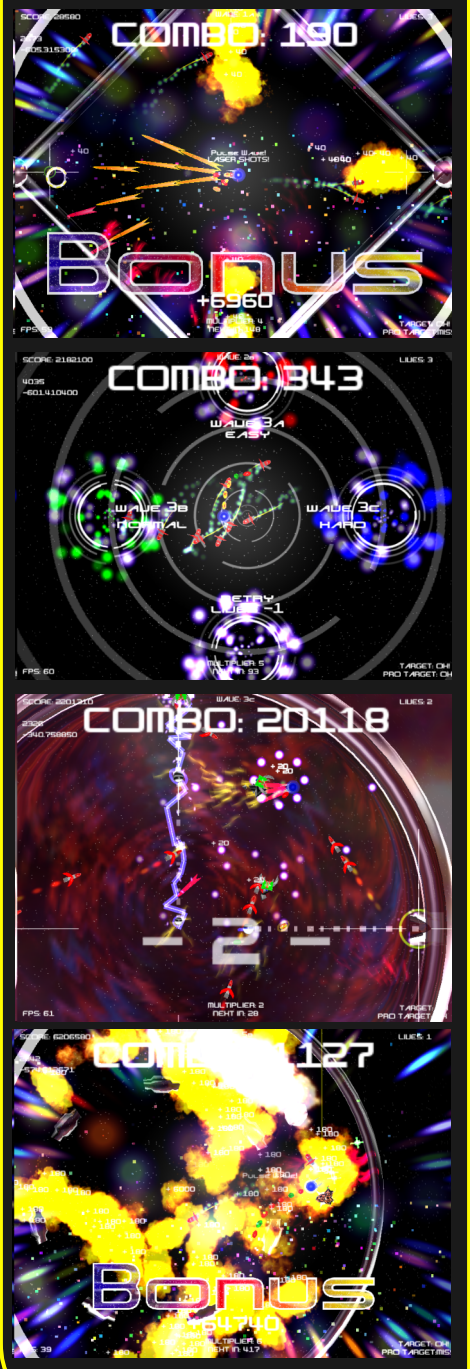
Charlie: Aside from the docs being a bit scant at times, I find the language pretty easy to use. The forums are great, people take the time to answer my stupid questions anyway...

Allan: It looks like you have moved to an Arena style playing area when in Bullet Candy 2 can I ask what brought about this change, and could you provide us with an overview of the other differences?

Charlie: A few reasons really. There are lots of arena games about now, and I want to make mine stand out a bit - hence the different shapes and styles of the play areas. Differences wise, where to start?! The original game had 50 levels, that you played through one after the other until you finished them all, I tried this again for BC2, but I got bored of it, it's what you expect, so now each level has multiple exits, which open or close depending on certain criteria. This way, you can't see everything the game has to offer on you first go (or you tenth or whatever), and allows players to choose a vast number of different routes through the various levels to find the one they like best/score highest/is easiest etc.. Replay-ability is important in arcade games, so coupled with fun gameplay, a fairly in-depth score mechanic and other cool stuff like achievements and skill tracking (think Wii sports-ish...) it think the finished product will have quite a bit to offer players, and that's not touching on the online stuff either... ;)

Allan: It sounds like Bullet Candy 2 is nearing completion how have you found the business and marketing side of being an indie games developer?

Project: Bullet Candy 2
Developer: jkrankie
Website: charliesgames.com



Charlie: It's getting closer every day, I hoping to have it on sale towards the end of the year. Once the game is done, there's all the fluff, like proper menus, all the database stuff involved in tracking statistics and achievements for each player and other extras like Xbox360 pad rumble support - and worst of all trying to make it run acceptably on crappy intel GFX laptops.

The business side of things has been a bit hit and miss for me, for every success (like Steam offering to sell my game for me :)) there have been at least double the amount of failures. For example, most casual portals I sell Bullet Candy

In the works...

through make so few sales that it hasn't really been worth the hassle. I'll certainly be putting this down to experience, and I won't be trying to sell BC2 through many of them, it just hasn't been worth the time it took to sort out contracts, incorporate various logos, remove parts of the game, add other stuff to the game etc.

On the marketing side, I'm totally hopeless :)

Allan: You seem to have developed a great game in a very short time can you give us an idea of the time you have spent working on Bullet Candy 2 and an idea of where the time was spent?

Charlie: Well, I work between 3 and 5 full days a week on the game, which always helps speed things along. To begin with, I mostly spent time getting the look and feel right. After that, probably getting the code structure right for main loops and base classes, then adding enemies, levels and fleshing out the basics of the gameplay so it starts to feel right. Most recently, it's been adding more levels and enemies that suit the theme of the level I'm working on and tweaking gameplay elements so that it plays better.

Allan: Once Bullet Candy 2 is complete what kind of games or technologies would you like to work on next?

Charlie: I've been toying with the idea of doing a platformer for a while now, so maybe I'll do one of those? I have a few ideas for one using some physics stuff, but in my previous experiments with physics engines I've found it tricky to make things fun, getting things to feel like a game rather than a simulation. I've got another project too that has been paused while I develop BC2, maybe I could go back and have another look at doing some more to that, or perhaps add the last two levels to Space Phallus...

Allan: Now you have the 20/20 visions that comes with hindsight what adivce would you give to other potential indie games developers?

Charlie: I don't know? Keep your targets within the realms of you ability I suppose, and make sure you don't release anything that isn't fun. Find some people whose opinions you trust to tell you how your game is going at various stages through development, it's far too easy to loose track of where things are heading of you don't let others play.

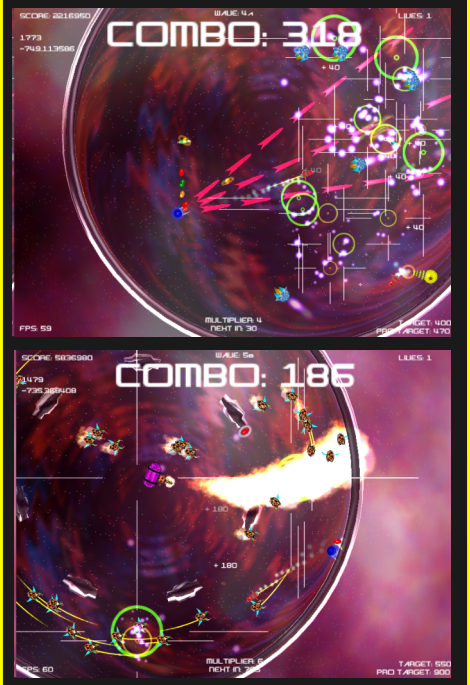
Allan: Would you be willing to provide BlitzMax Coder with a Postmortem article covering the technical aspects of your Bullet Candy 2, What went right, What went wrong and what you would have done differently?

Charlie: I suppose so :)

Allan: Thank you and the best of luck with Bullet Candy 2, if you would like BlitzMax Coder to review the game let me know! ;0)

Charlie: No problem :)

Project: Bullet Candy 2
Developer: jkrankie
Website: charliesgames.com



In the works...

Have you ever played the vector based arcade version of the Star Wars or Empire Strikes Back what about the vector version of Return of the Jedi? No... well read on!

Allan: Could you give the readers a brief outline of your background and what brought you into computer games development?

Andrew: Where to start? I guess I became interested in computers when I was about ten years old. My dad bought a ZX81. He didn't really get much of a look in because me and brother were always on it playing Mazogs. We upgraded to the ZX Spectrum a few years later and then work our way through various systems. I tried programming back then in Basic. I used to type in games printed in magazines and tried to figure out how they worked.
Eventually I preferred playing games than trying to write them. My interest in computers waned when consoles took over as being the main platforms for games.

Skip forward to 2002, whilst surfing the web I came across a copy of Klass of 99 which lead me to the Retrospec and Retro Remakes website.

Here I found all the games I loved as child updated for the PC. I thought this was a brilliant idea. So I decided to learning programming and remake one of my favourite childhood games 3D Star
Strike for the ZX Spectrum. Initially tried several languages (C++, Dark Basic) but finally settled on Blitz3D as it was the best fit for me. It took me 5 years to finish the game I first started. It became a labour of love, but I was pleased with the results and certainly learnt a lot along the way. I made another two games since then.

Allan: Hi Andrew, your developing a 'Remake that never was...' of Return of the Jedi - in a retro vector style can I ask what triggered the idea?

Andrew: The idea formed over a few months. It wasn't a sudden realization, more like several ideas / inspirations came together at once.

About a year ago a fellow Blitz member JoeGr posted a demo of his vector engine in Blitz3D (http://www.blitzbasic.com/Community/posts.php?topic=74505#832888). I could immediately see this could be used to recreate or make new vector games.

Joe never released his source, but my interest was piqued so I decided to try and write my own version in Blitz3d. Joe was kind enough to give me some pointers, along way and I finally made a working version (http://www.blitzbasic.com/Community/posts.php?topic=76935#861121). This sat on my hard drive whilst I tried to work out what to make with it first.

In the meantime I had always loved the original Star Wars Arcade game by Atari. Whenever I could I would get my brother to drive me down to Barry Island to play it. They had a nice Cockpit version, and it was totally immersive with the sound was cranked up and the flying the
x-wing intuitively with the yoke control. I guess this left an lasting impression on me, hence me remaking 3D Star Strike which is a clone of this

Project: Return of the Jedi
Developer: Andrew
Website: urbaninteractive.com

In the works...

game.

Atari made the complete first Star Wars trilogy. Star Wars and Empire Strikes back featured vector graphics, but the Return of the Jedi was an isometric raster graphics scroller which didn't really play very well. I always thought this was a missed opportunity as the Battle of Endor sequence would be great fun to play in first person.

Finally the idea came together when Retro remakes held a competition and one of the categories was "Sequels that never were...", this got me thinking why not make a remake ROTJ as I would have liked it, using the engine I had developed.

Allan: But the project is more than just the game could you tell us more about the scope of the project and what you aim to achieve?

Andrew: Yes, the scope of the project did change. It kind of took on a life of it's own. Initially I was just going to make the game for the PC, and I started a blog to cover its development (www.urbaninteractive.wordpress.com). Soon some people came across the blog and links to started appearing on a few MAME websites. I was really surprised by the level of interest shown in it. Soon I had lots of ideas flooding in from other people. Some MAME users wanted the game to put inside their arcade cabinets. That's when I thought wouldn't be cool to also make a cabinet to house the game in. I spoke to my wife and she didn't seem to mind as long as it stays in office once built. So far I have built about half of the cabinet, and have gathered all the pieces needed to finish it. I have an original Star Wars yoke as well which I had to hack to make it work with the PC. It really adds to the game playing with this controller, however the final game though will also allow mouse and joystick control.

Allan: I notices that quite early on you moved over to BlitzMax porting from Blitz3D to BlitzMax, how did you find the transition?

Andrew: Yeah I decided to move it over the BMax, because I had recently become a convert to OOP after many years not understanding all the fuss was about. Also the cross platform feature was an added incentive. The migration of the vector engine took about a month with some tweaks still being applied. The code is much neater now using OOP. For the graphics engine initially I used the B3D.dll, but finally opted for MiniB3D in the end as it gave me more control i.e. I could now even edit the under laying graphics engine to suit my needs. All in all the migration was pretty painless actually.

Allan: ROTJ sound like quite a big undertaking can you give us an outline of the the development tools or pipeline you use from idea through to realisation?

Andrew: So obviously I'm using BMax, and MiniB3D. The modelling pipeline isn't too long. To create the base model I use Silo. I really love the interface, and the functionality of this modeler. It's so easy to use.

Once the models are made they converted to B3D format using Ultimate Unwrap. They are then loaded into a little editor I wrote using MaxGui and MiniB3D, which allows me to select the edges of the mesh that will be drawn by the vector engine. I can select the colour, the alpha value and set special attributes per edge e.g. attribute 2 is flash random colours. The editor then save a file which can be loaded by the vectore engine. I also have another editor for create 2D vectors for the fonts etc.

Allan: Everyone has the favourite moments for the STAR WARS movies what would you say is yours from Return of the Jedi?

Andrew: Pretty much any scene without an Ewok is good in my book. I guess the whole Battle of Endor sequence would be my favourite. I hope to recreate some of that action and pace in my game.

Allan: George Lucas went prequel with a follow on to the STAR WARS trilogy have you thought about doing the The Phantom Menace, Attack of the Clones, Revenge of the Sith in a classic vector retro style?

Andrew: Honestly the prequels didn't grab me in the same way as the original movies. For sure I have seen them and they complete the circle, but I prefer it when all the questions aren't answered.
Therefore I don't think I would make any games based on the prequels. By the end of this project I think I would have finally put to rest my desire to make Star Wars inspired games.

Allan: As ROTJ is a Remake that never was could you give us a rough idea of the levels/stages you would like to include in it?

Andrew: Currently I am toying with having three levels but this may change to four later. The first level will The Battle of Endor, where the player will pilot an X-Wing. It will be similar to the first level of the first Star Wars game, except there will be a lot more going on. The premise of this level is that you have been assigned as wing support for the Millennium Falcon on it's mission to blow up the Imperial Star Destroyers. You will follow the Millennium Falcon whilst it weaves its way through the Rebel Alliance Fleet on its way to the Star Destroyers. The Millennium Falcon will only be able to take a certain number of hits from the enemy, so the player will need to ensure it picks of any incoming plasma bolts and enemy craft before they hit it.

The second level is the speeder bike level. Here the player will be able to whiz through the trees picking off the enemy on the way. At the end they will reach the doors of the shield power generator which will then be disabled allowing access to level 3.

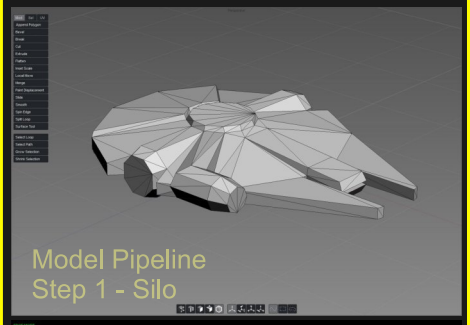
In the final third level the player will fly the Millennium Falcon across the surface of the Death Star before heading through the catacombs of pipe work to the central core. The is will be similar to the trench run in the first movie, but the play is to include twists and turns in the tunnels, and to have enemy fighters also within the tunnels.

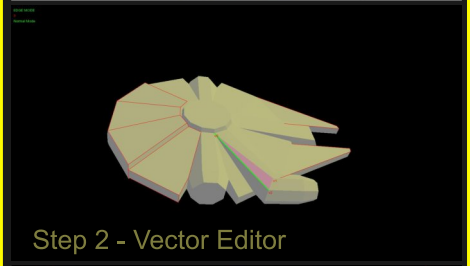
Allan: Given the experience you have gained so far in games development what advice would you give to other potential game developers?

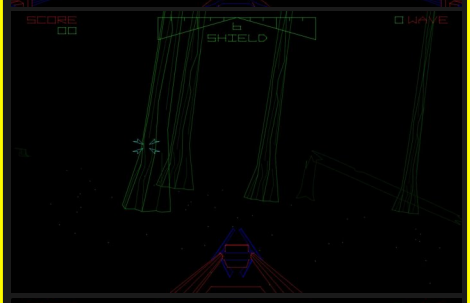
Andrew: It depends on your level experience really. If you are completely green to programming then I would suggest starting with something very simple like space invaders. After all these type of games are where the industry started. I lot of people start with unrealistic targets like making a MMORPG, and get disheartened when it doesn't come off. So I would suggest start small, and build up experience. Focus on the basics, and get that working then add all the bells and whistles at the end.

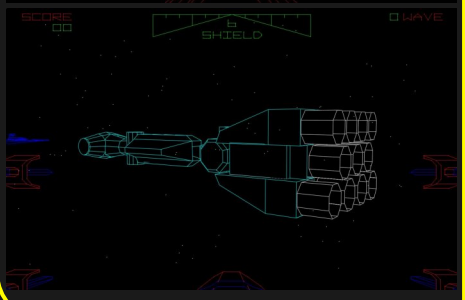
Also I would suggest having a clear objectives. In my first game I was forever adding extra bits here and rewriting bits there and that's why it took so long to make. With this game I know what I want and once I achieve those targets for me the initial

Project: Return of the Jedi
Developer: Andrew
Website: urbaninteractive.com









In the works...

version will be finished and released.

I think the biggest advice is if you are coding for a hobby then enjoy it :)

Allan: How have you found devleoping in the BlitzMax language overall?

Andrew: Initially I found BMax a little daunting, especially with the limited documentation but this is where forum really comes into its own. For nearly every question I had a quick search on the forum provided me with an answer, or at least a clue of what to do next. I now a complete convert.
I find the syntax good, and the idea of modules very handy. Developing in BMax has also improved my coding because I have adopted an OOP approach. My code is also split into modules and it's much easier for me to follow, modify and maintain. I also use Super Strict mode which forces me be accurate with all my declarations. Generally I have enjoyed programming in BlitzMax, and it's now my language choice.

Allan: Do you have any other games/ideas that you would like to develop once ROTJ is complete?

Andrew: Yep, I have a big long list of games I would like to remake. However I think I would like to make an original title of my own, and probably in 2D next.

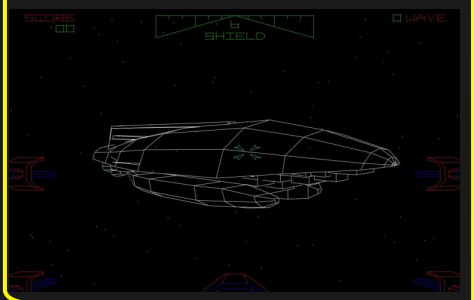
Allan: Would you be willing to provide us at BlitzMaxCoder.com with a Gamasutra style post mortem of the game when you have completed it?

Andrew: Absolutely. I hope BlitzMaxCoder is a success.

Allan: Thank you for contributing to BlitzMax Coder and as a fan of the vector STAR WARS consoles I would like to wish you good luck and please let us know when it's complete?

Andrew: Sure, and for those who are interested you can follow the progress at my blog or over at Retro Remakes.

Project: Return of the Jedi
Developer: Andrew
Website: urbaninteractive.com

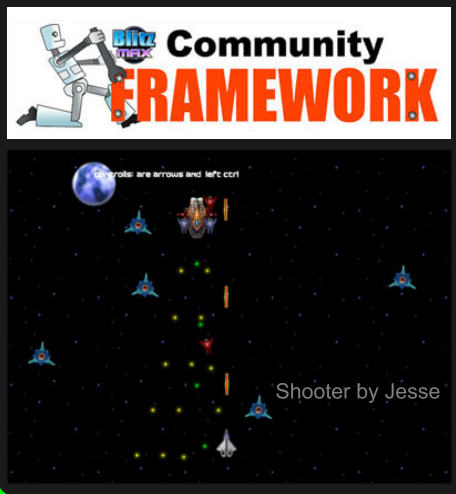


Interview

In this interview with Pete, we are going to find out about the BlitzMax community framework, note that the framework is still a work in progress and is gradually building up to provide features that will allow developers easily and quickly produce quality games and applications with BlitzMax.

Allan: Hi Pete, could you tell us a little bit about yourself and what brought you to developing with BlitzMax?
Pete: Hi Allan, I'm a 30 year old Sound engineer from Somerset England, I got started programming with Blitzbasic on the Commodore Amiga. When I sold my Amiga I did very little programming, spending most of my time creating graphics. About 5 years ago I really wanted to make a game idea Id had for a few years, So on searching the net I was very happy to find that Blitz was available for the PC. Although I've been programming for years now its only in the last year or so that I've made the jump to object oriented programming. I really found programing hard to learn I'm in no way a natural.
Allan: How have you found developing in the BlitzMax language?
Pete: I think apart from Blitzmax and Blitz3d which are great languages to start with, The best thing about Blitz is the community, There are lots of great experienced blitz programmers that are only to happy to help with any problems or questions you have. You only have to look at our list of sponsors to see how great the Blitz community is. Ive programmed in Java and C#, but I still come back to Blitzmax, I just love it.
Allan: You have launched into a large community project that aims to provide a framework for developing games in BlitzMax, what was the inspiration that brought this about?
Pete: The Framework was born from a discussion on the Blitz forums, Noel(Chroma) suggested a community game project. I thought that only a small number of people would want to make the same type of game and it would be better to make something everyone could use, how and when they liked. I thought why not try and get something going and set up a website and try and drum up some interest. The reaction from the community was great and with Kennie (Pharmhaus) taking charge of the updates and edits to the code a it came in, the whole think has grown really fast.
Allan: Could you give us a breakdown of where the community framework is at the moment and what features it provides?
Pete: I look at the framework in two parts, First we have the accentual but boring code, That every program needs to be of a professional standard. These include: Delta timing (display.mod) This makes the game run consistently across different computer. Also code for making windows minimise and close correctly. You get the idea. Second We have the more advanced modules, like a Particle engine and another that handles advanced OpenGL functions. Math.mod which handles a lot of mathematics useful when dealing with movement. There are far to many to list and we are adding more all the time. There's still lots from the last competition we still need to add. I'm working on a guide at the moment that can be added to as we go and will have an online and pdf version available. We are working hard to reshape and tidy up the modules to make everything easy for the people downloading.
Allan: What features would you like to see in future versions of the framework?
Pete: I've got a few ideas of what I'm going to add once I've finished the guide but I think things are developing nicely, a Physics or 3d engine would be great, but lets see what happens in the next compo. One thing we do need, is more good examples of the framework functions in use.
Allan: Can anyone contribute to the framework, are their guidelines for contributions?
Pete: Anyone who has Blitzmax can contribute. The only guidelines are to keep your code as readable as you can, make it as OO as possible for easy integration and of course no copyrighted material.
Allan: You have already had one competition with prizes, that has raised the profile of the community, how did that go?
Pete: Really well. As it was our first Competition I think we made a few mistakes with the timing, but I couldn't really ask for more, we got some great code from it and everyone had lots of fun entering.
Allan: Your second competition has launched recently with again a load of sponsored prizes and the theme is 'Eye Candy' are there any particular areas of the Framework you are hoping to see utilised and expanded upon?
Pete: Well you know how everyone loves eye candy, I just though It would be fun and attract more people to try the framework, if

Project: BlitzMax Community Framework
Developer: Pete Carter
Website: blitzframework.co.uk



Interviews

only to see how some of the entries were done. I'm also hoping to see entries that use some of the graphics functions we already have to create great examples of how to use the framework.

Pete: Well, testing and bug finding is needed, I really hope to get a health number of forum users as feedback and ideas are always welcome. I want to make this framework useful for everyone from beginners to more advanced coders.

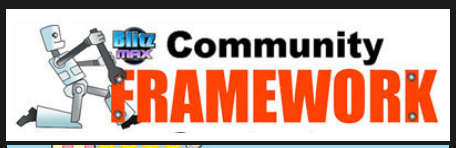
I have a massive amount of Maths functions to add. A lot of code that has been submited as useful functions, but in no really order and a better GUI library.

There are a number of modules that are really useful in the current release.

Current BlitzMax Community Framework Modules in alphabetical order:

Module name	Description
Audiohelper	This adds a few useful functions to the simple audio commands in Blitzmax.
GetSampleLength	Determines length of an audio sample.
oggToWave	Convert an ogg to a wav file.
Playsoundbetter	Play sound with pan and spread at a given location.
Bitmapfont	Bitmap font type.
Compression	Compression library, compresses/decompresses resources.
Core	Imports all of the modules in one line of code. For easy setup.
Display	Handles rendering to the screen including delta timing for consistent time across different systems. Plus a FPS counter.
Entity	An entity system much like Blitz3d. Allows objects to be parented to over objects.
Event	Event management. Sets up a system to control you program using events as triggers.
GUI	Create drop down menus, buttons etc for your apps.
Lite	Allows you to save all your game information to a file.
Managed ram	Manages your ram, loading and clearing as required.
Maths	Lots of maths functions.
Particles	A Particle engine.
Scope	Controls rendering of objects to within a set area (scope).
GLhelper	Lots and lots of Open GL functions.
Maxml	Loads and saves XML data.

Project: BlitzMax Community Framework
Developer: Pete
Website: blitzmaxframework.co.uk



Competitions

Prize Sponsors

Arteria
Whitegatesoftware.com Software Protection
KORIOLIS FX
uSprite II
Basilisk Games
Tank Universal
Grey Alien Games
BLIde Enjoy development
XNET
Ancient Soft
IMPHENZIA.COM
D-GRAFIX
Logic Zone
Charlies Games

The Blitzmax Community Framework Team is proud to present their second blitz competition.

If you like to join the contest, please visit the BlitzMax forum for details.

All you have to do is write a game which fits the theme and post it before the deadline has passed.
Competition Rules (The judges will score on):

- OOPness of your code
- Impressiveness
- Cleanness of code
- And for sure Fun :-)

Once the deadline has passed the judges will decide the winners of the 1st , 2nd and 3rd prize.

The Winner having first choice of the prize they would like, once the 1st prize winner has chosen the 2nd prize winner chooses a prize and so on.

All three winners of this contest will get a free copy of *Bullet Candy* as an extra prize!

Contest: BlitzMax Community Framework Contest 2
Runner: Pete
Theme: Eye Candy
Start: 1 July 2009
Finish: 1 August 2009
Website: blitzmaxframework.co.uk

Feedback

Project:	BlitzMAX coder Feedback
Level:	Any
Author:	Anyone
Email:	feedback@blitzmaxcoder.com

That's all folks...

What you wanted to see more, or you didn't like it, well then why not contribute or provide feedback to BlitzMAX coder...

I would like to thank everyone who has contributed to the very first issue and hope you will continue to contribute to this new magazine.

Apologies if your article or feature did not make it into this edition but be assured it will appear in the next issue.

I would especially like to thank the BlitzMax forum members who commented on my original logo design, without your input the magazine's logo would have looked like this...

which prompted the following feedback from various forum members:

> ACiiiiiiiiiiiiiiid.....ACIIIIIIIIIIIIID! :)

> It reminds me of hippies.

> My opinion only, the logo (and the site), look like somebody ate all of the characters on Sesame Street and then puked them up.

> It is like a Dali painting. It grows on you the more you look at it. I am slowly coming around to the pro muppet puke side.

> Looks well made, but I think it breaks every logo design rule there is :)

So think yourselves lucky you it could have turned out a lot worse... anyway what next this is just issue one after all...

Well the supporting BlitzMax coder website will follow 'shortly' but I will probably play with some ideas for what should be included and then run them past you in the forums.

Things I would like to add in future issues...

- A breakdown of all of games and applications made with BlitzMax.
- Reviews of BlitzMax games and applications.
- A catalogue of all of the modules, frameworks, and code archives.
- More Interviews - what are you doing with BlitzMax
- More Articles - Games Development, Programming, Design...
- More Tutorials - How To's, guides

in effect a a good companion site to the forum but let me know what you would like to see.

Kind Regards

Allan (aka Merx)

blitzmaxcoder.com

arowx.com