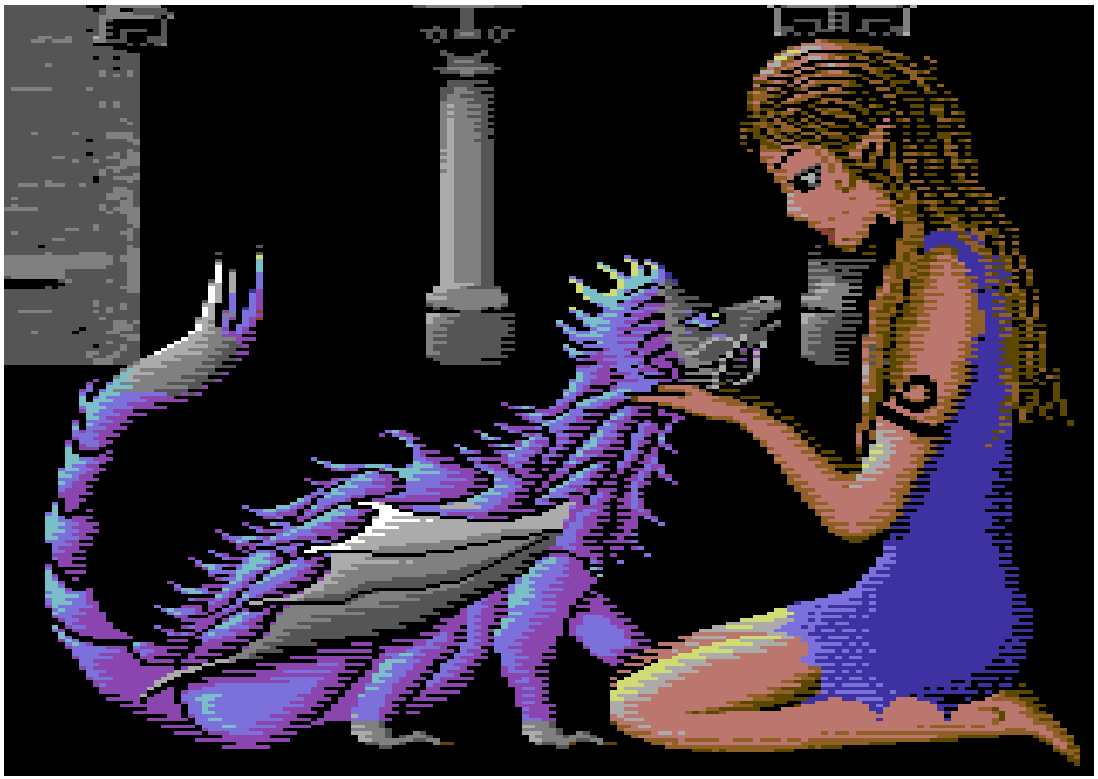
Making of "The Dragonbond"

German: Page 10 - 18
English: read here

Endpart: „Fairytale“



Story and Concept:
I wanted to tell a story on the C64 – a fairy tale, a kind of animated picture book.
It was meant to be entirely hand-drawn in Pixcen, in the style of French comics.
The fairy tale tells the story of an elf named Leandra, who, in youthful recklessness, saves a young dragon – even though she should have known that it was forbidden for good reason.
In this world, dragons form a bond: they are absolutely loyal and faithful, but if they are hurt, they become wild beasts – dangerous and uncontrollable.
To Leandra, the dragon is just a good friend, but what she doesn’t know is that the dragon would never accept the elf having another friend. But then Leandra falls in love with another elf, and fate takes its course.

The dragon grieves, feeling misunderstood and rejected. He turns into a beast and takes revenge on mankind. Only the elf herself can kill the dragon – no one else.
In the final scene, she kneels sorrowfully before the dragon – her friend – whom she had to kill with her own hands.

The graphics are based on pencil sketches by my daughter Elisa and were then redrawn in Pixcen.

Technical Implementation:

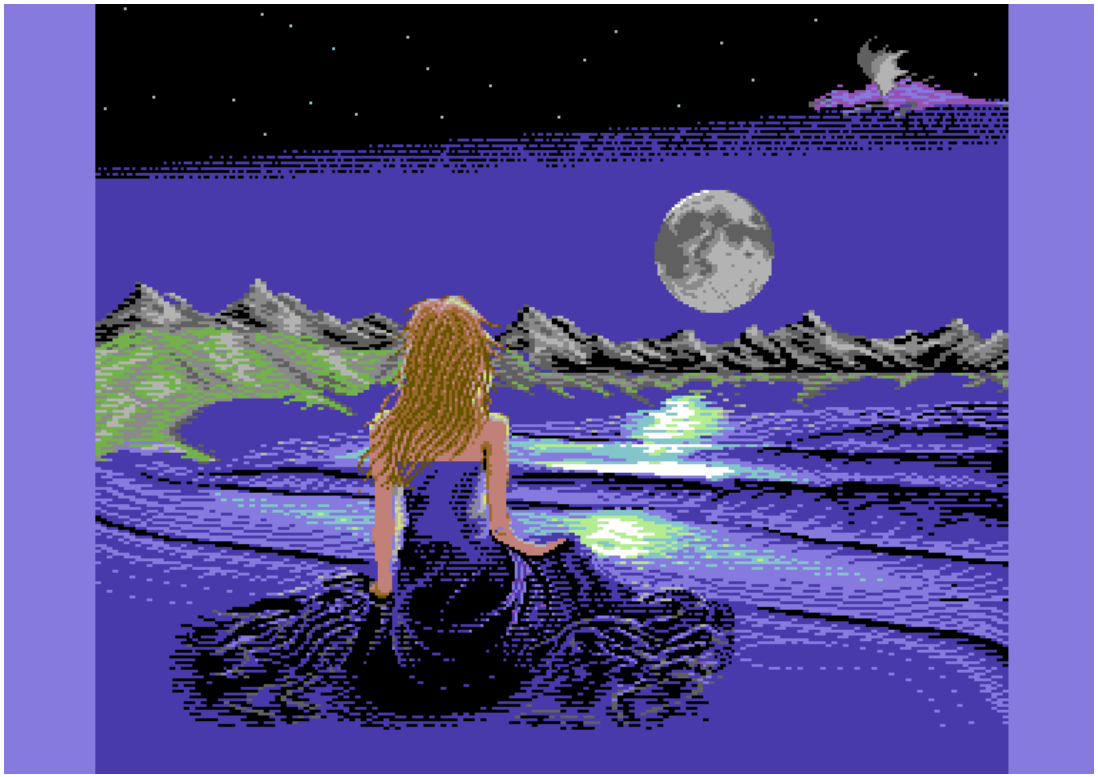
The story was realized in bitmap mode, using double buffering and both horizontal and vertical scrolling. The goal was to give the impression of a large poster being filmed through camera movements.

(zoom in)

Graphic data are loaded and decompressed in real time using Bitfire (by Bitbreaker), alternating between two 1K-byte blocks. To enable horizontal and vertical scrolling, the screen must, as is known, be reduced to 24 rows and 38 columns.

To avoid unnecessarily lowering the resolution, static images are displayed in full resolution, and the row and column switching is smoothly faded in and out through animation, depending on the scene's needs. In some cases, single-color sprites were overlaid on objects when the multicolor resolution was too coarse. Otherwise, sprites were only used to add small animations.

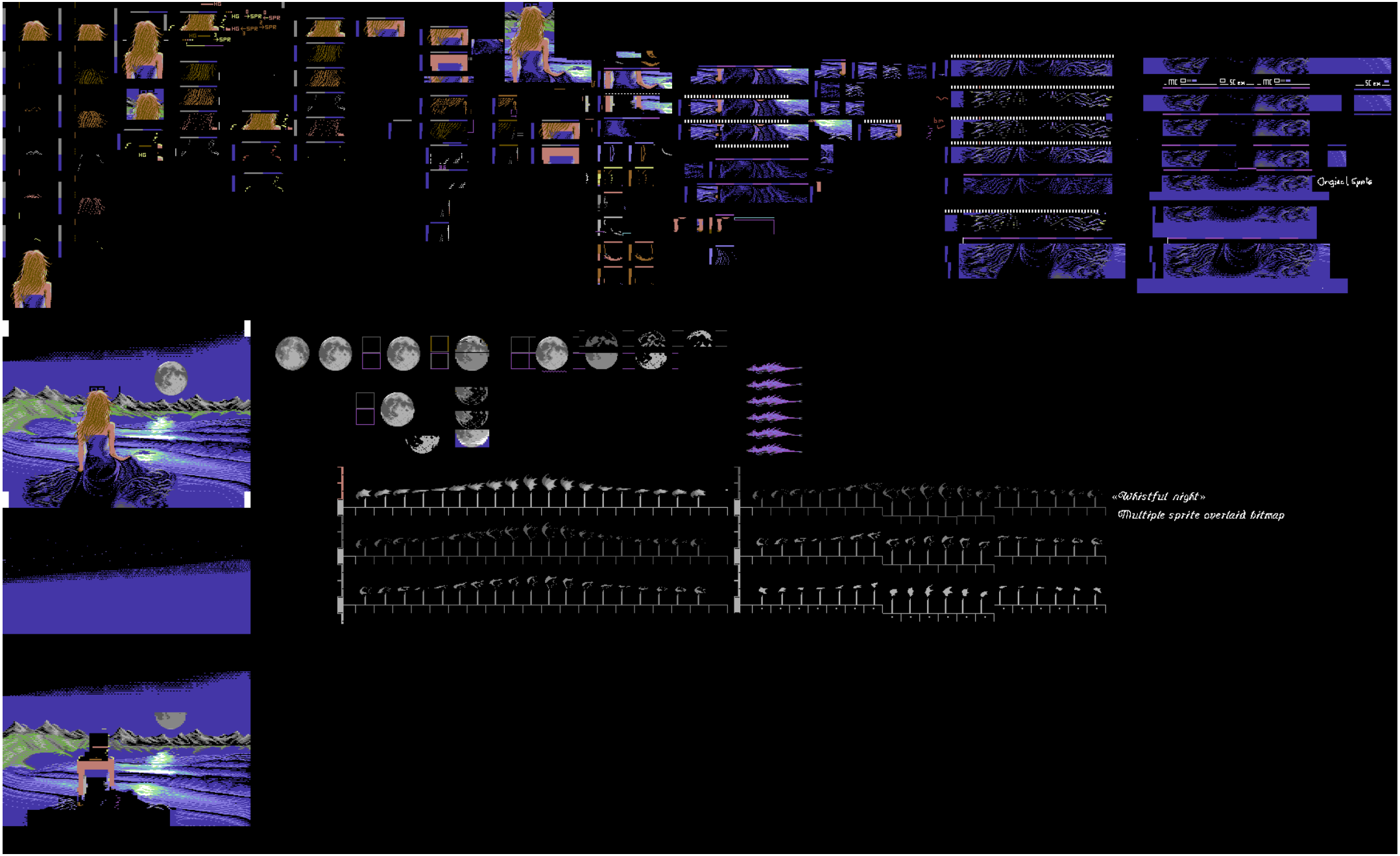
Intro Part: “Whistful Night”



Story and Concept:
The intro part deals with Leandra’s youthful and naive wish to have a dragon of her own.
She waits by a mountain lake in the moonlight, hoping at last to see a dragon in the sky.

Technical Implementation
First, I drew the elf and the moon in hires with 16 C64-colors using Pixcen, without considering whether the C64 would actually be able to display the image.
The background (lake, mountains, etc.) was created in C64-compatible multicolor mode.

In the second step, the necessary overlay sprites were created based on the concept graphic.
The elf consists of a combination of a few bitmap pixels and an overlaid flat sprite.
In almost every rasterline where the elf or the moon appears, all eight sprites are in use.
From the moon down to the elf, a matrix of 64 sprites is used in total.


Original Sprite
«Whistful night»
Multiple sprite overlaid bitmap

Unfortunately, the available sprites are not sufficient to reproduce the entire image exactly as in the original concept.
However, it is possible to position the dots differently than in the concept file — for instance, a double-width bitmap pixel can be visually halved by overlaying it with a single-color sprite pixel. This creates the impression of higher resolution.
When creating the overlay sprites, it is necessary to proceed point by point and line by line, repeatedly finding smart individual solutions for each problem — figuring out how to place a "difficult" pixel in one way or another.
Often, it is possible to match the original almost exactly, but occasionally small deviations are necessary.
In some cases, "difficult" pixels can only be displayed by splitting sprites horizontally to a different X-position during rendering. The corresponding sprite bit patterns then have to be adjusted accordingly.
In some spots, both a sprite-X-split and a color adjustment are required to display "difficult" pixels in the correct color.
The lower border is rendered with a mix of multicolor sprites and expanded single-color sprites due to the limited number of available sprites. Single-color sprites are well suited for expansion when extending the multicolor background graphics into the lower border, since they share the same resolution. Register $d021 was also adjusted.
At rasterline 0, the bitmap mode switches to high-resolution in order to display the color transition (above the moon) with greater detail. Only afterward does the multicolor bitmap with the mountains and lake begin.
All of this was very time-consuming, but fascinating — there's always some kind of solution to be found.
Since many registers need to be changed within a very short time at the lower edges of the sprites to trigger the next sprite split, I chose the following approach, which is less timing-critical:

Example (simplified, using only one sprite):
Let's assume Sprite 0 is currently displayed at Y-position 100 with Sprite Pointer 0.
At Y-position 121, Sprite 0 should be displayed again, but with a different sprite pointer — for example, Sprite Pointer 16.

Solution ("intermediate sprites"):

1. Starting from Y-position 101 of the sprite (that is, when the raster beam is drawing the second line of the sprite), I already set the new Y-position (121) for the next sprite split. The VIC executes this only after the current sprite has been fully drawn — that is, at Y-position 121.

2. Starting from Y-position 110 (when the raster beam is drawing the 11th line of the sprite), I switch to another sprite pointer. In this sprite matrix, bytes 0–29 belong to the new sprite (which will later be displayed at Y-position 121 [Sprite Pointer 16]), while bytes 30–62 still belong to the old sprite (which is currently being displayed at Y-position 100 [Sprite Pointer 0]). In other words, I work with "intermediate sprites." This intermediate sprite could, for instance, be stored under Sprite Pointer 8.

3. At Y-position 121, no further action is required — thanks to these "intermediate sprites". I now have additional processing time to adjust the X-position or colors, if needed.

4. Starting from Y-position 122 (that is, when the raster beam begins drawing the second line of the new sprite), the complete new sprite must be activated, since the matrix of the intermediate sprite only covers up to sprite line 11.

For the register changes described in steps 1, 2, and 4, I have plenty of time — up to nine raster lines.

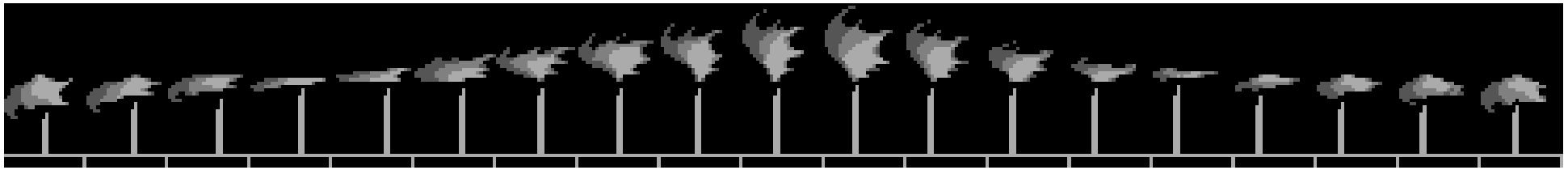
Flying Sprite Dragon

The dragon's body consists of only three animated multicolor sprites.
The wings are made up of three overlapping single-color sprites to simulate high-resolution multicolor. In total, 57 sprites were used for the wings across 19 animation frames.
The wingbeat of a bald eagle served as a reference.

My daughter Elisa designed a drangon wing model out of cardboard, which was made movable with solder wire.

Using a tablet, the wingbeat was recorded as stop motion with the involvement of my daughter Julia – though only with 11 animation steps

The photos were converted to C64 grayscale, refined in Pixcen, and additional intermediate frames were created to make the animation smoother.
To make full use of the 24×21 pixel matrix of each sprite, the sprite was adjusted in both X and Y directions during animation (using a compensation table). The rod holding the cardboard wing served as a reference for positioning the sprites along the X and Y axes.



(zoom in)

Starry Sky
The starry sky in the upper border consists of a single sprite that is split every two raster lines to a different X-position.
Below it, the stars are drawn in the bitmap area, which is already set to hires for the color transition.

Overall, two VIC banks had to be used for this part because the sprite memory requirements were too high.

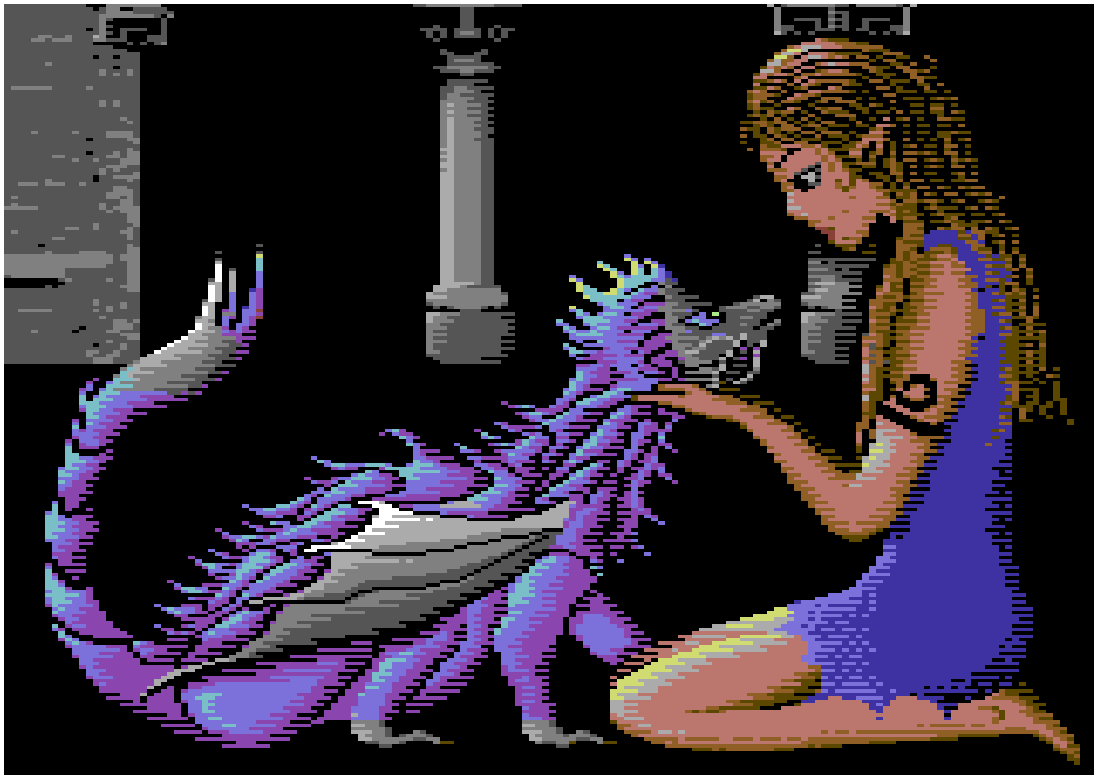
Many thanks to Elisa, Julia, Carmen! And of course, Abyss-Connection!

Abyss-Connection: Living for the Cursor

Making of „The Dragonbond“

(Deutsch)

Endpart: „Fairytale“



Story und Idee:
ich wollte auf dem C64 eine Geschichte erzählen, ein Märchen, eine Art animiertes Bilderbuch.
Es sollte vollständig mit Pixcen im französischen Comic-Stil handgezeichnet sein.
Das Märchen erzählt von der Elfe Leandra, die in jugendlichem Leichtsinn einen jungen Drachen rettet,
obwohl sie wissen musste, dass es aus guten Grunde verboten ist.
Denn in dieser Welt gehen Drachen einen Bund ein: Sie sind absolut treu und loyal, aber wenn sie gekränkt sind, sind sie wilde Tiere und gefährlich. Für Leandra ist der Drache nur ein guter Freund, doch was sie nicht weiß: der Drache würde niemals akzeptieren, dass die Elfe einen weiteren Freund hat. Doch nun verliebt sich Leandra in einen Elfen und das Schicksal nimmt seinen Lauf.

Der Drache trauert, fühlt sich unverstanden und zurückgestoßen. Er wird zur Bestie und rächt sich an den Menschen. Nur die Elfe selbst kann den Drachen töten, niemand sonst. In der Endszene kauert sie traurig vor dem Drachen, ihrem Freund, den sie selbst töten musste.

Die Grafiken basieren auf Bleistift-Skizzen meiner Tochter Elisa und wurden darauffolgend mit Pixcen gezeichnet.

Technische Umsetzung:

Technisch umgesetzt wurde die Story mit Bitmap, dass im im Double-Buffer, horizontal und vertikal gescrollt wird. Es soll der Eindruck entstehen, das es sich um ein großes Plakat handelt, dass durch Kamerabewegungen, abgefilmt wird.

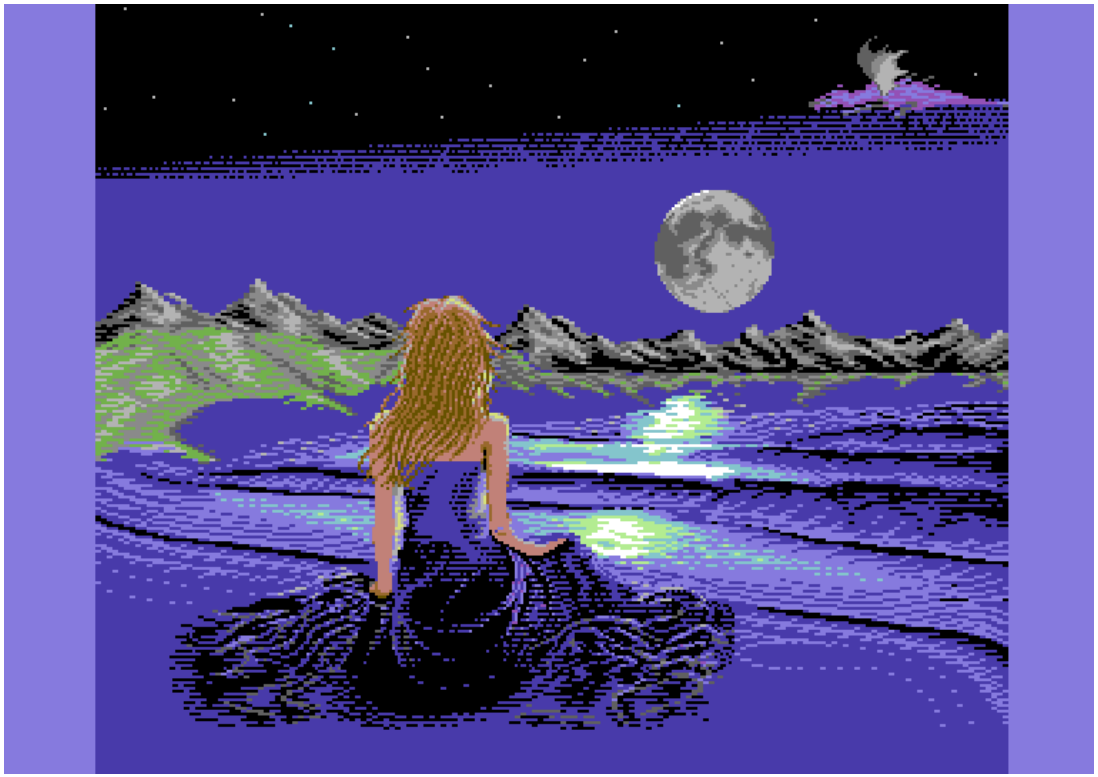
(zoom in)

Grafikendaten werden in Realtime mit Bitfire (Bitbreaker) abwechselnd in zwei ein KByteblöcke nachgeladen und dekomprimiert. Um horizontal und vertikal scrollen zu können, muss der Bildschirm bekanntlich auf 24 Zeilen und 38 Spalten reduziert werden.

Um die Auflösung nicht unnötig zu verringern, werden stehende Bilder in voller Auflösung dargestellt und die Zeilen- und Spaltenumschaltung je nach Notwendigkeit durch eine Animation möglichst weich ein- und ausgeblendet.

Z.T. wurden Objekte mit Single-Color-Sprites überlagert, wenn die Multicoloraufösung zu grobkörnig war. Ansonsten wurden Sprites nur verwendet, um kleine Animationen einzubauen.

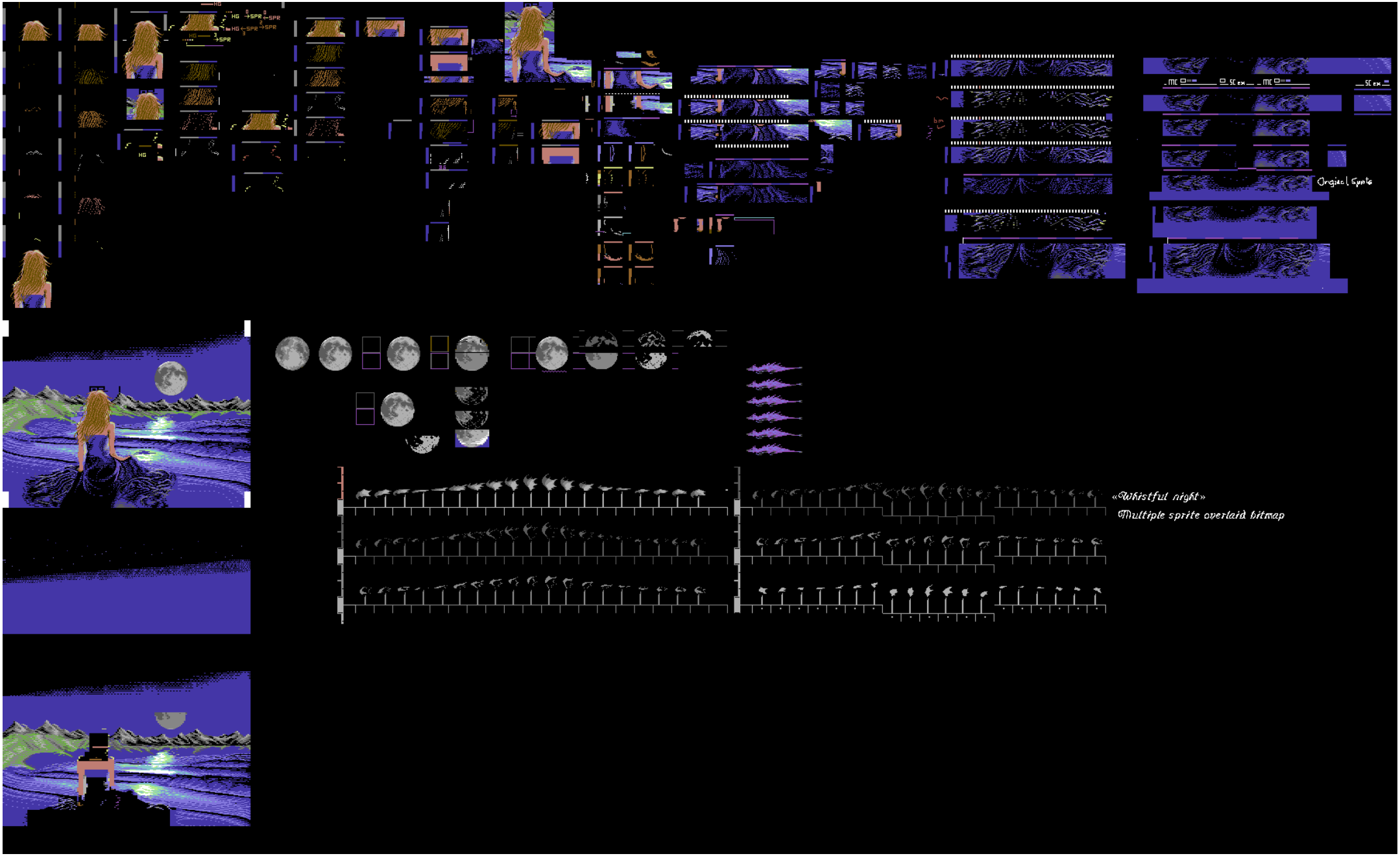
Vorpart: „Whistful night“



Story und Idee:
Im Vorpart wird die jugendlich-naive Sehnsucht Leandras thematisiert, einen Drachen zu besitzen.
Sie wartet im Schein des Mondes an einem Bergsee, um endlich einen Drachen am Himmel zu sehen.

Technische Umsetzung:
Zunächst habe ich die Elfe und den Mond mit Pixcen hochauflösend mit 16 C64-Farben gezeichnet und keine Rücksicht darauf genommen, ob der C64 das Bild überhaupt darstellen kann. Der Hintergrund (See, Berge usw.) wurde C64-konform im Multicolor-Mode gezeichnet.

Im zweiten Schritt wurden die notwendigen Overlay-Sprites aus der Konzeptgrafik erstellt. Die Elfe besteht aus einem Mix aus wenigen Bitmap-Pixeln und einer flächigen Spriteüberlagerung. Es werden in jeder Zeile, in der die Elfe oder der Mond dargestellt werden, fast immer alle 8 Sprites benötigt.


Original Sprite
«Whistful night»
Multiple sprite overlaid bitmap

Insgesamt wird von Mond bis Elfe eine Matrix von 64 Sprites verwendet.
Leider reichen die Sprites nicht aus, um alles so darzustellen, wie es auf der Vorlage ist.
Es ist aber möglich, die Punkte anders zu setzen als auf der Konzeptvorlage, sodass ein bitmap-doppelt-breiter Punkt durch einen überlagerten Single-Color-Sprite-Punkt halbiert wird.
Damit erscheint alles hochauflösend.
Man muss sich bei der Erstellung der Overlay-Sprites jedoch von Punkt zu Punkt und von Zeile zu Zeile durcharbeiten, um immer wieder eine intelligente Einzellösung für jedes Problem zu finden – eben, wie man den einen oder anderen „schwierigen" Punkt doch irgendwie setzen kann.
Es ist häufig möglich, die Punkte so wie in der Vorlage zu setzen, aber manchmal muss man geringfügig von der Vorlage abweichen.
Manchmal lassen sich „schwierige" Punkte nur darstellen, wenn man Sprites während der Darstellung auf eine andere X-Position splittet. Die Sprite-Bitmuster müssen dann entsprechend angepasst werden.
An einigen Stellen müssen zusätzlich zum Sprite-X-Split die Spritefarben angepasst werden, um „schwierige" Punkte in der richtigen Farbe darzustellen.
Der Lower Border ist aufgrund von Spritemangel mit einem Mix aus Multicolor-Sprites und expandierten Single-Color-Sprites dargestellt.
Single-Color-Sprites eignen sich zur Expansion, wenn man die Hintergrund-Multicolor-Grafik im Lower Border weiterführen möchte, da sie die gleiche Auflösung haben. $D021 wurde zudem angepasst.
In Rasterzeile 0 wird das Bitmap auf Hires umgeschaltet, um den Farbübergang (oberhalb des Mondes) hochauflösender darzustellen. Erst danach beginnt das Multicolor-Bitmap mit Bergen und dem See.
Alles sehr zeitaufwendig, aber spannend – irgendeine Lösung gibt es immer.

Da insbesondere an den Unterkanten der Sprites viele Register in kurzer Zeit geändert werden müssen, um den nächsten Sprite-Split einzuleiten, habe ich folgenden Weg gewählt, der weniger zeitkritisch ist:

Ein Beispiel (um es zu vereinfachen, nur mit einem Sprite):
Ich stelle gerade Sprite 0 auf Y-Position 100 mit dem Spritepointer 0 dar.
Auf Y-Position 121 soll wieder Sprite 0 mit einem anderen Spritepointer dargestellt werden, z. B. mit dem Spritepointer 16.

Lösung („Intermediate Sprites“):

1. Ab Y-Position 101 des Sprites (gemeint ist, wenn der Rasterstrahl die zweite Zeile des Sprites darstellt) setze ich bereits die neue Y-Position (121) für den nächsten Y-Spritesplit. Der VIC führt das erst aus, nachdem die Sprites vollständig dargestellt sind – also erst an Y-Position 121.

2. Ab Y-Position 110 (gemeint ist, wenn der Rasterstrahl die elfte Zeile des Sprites darstellt) schalte ich auf einen anderen Spritepointer. In der Spritematrix sind die Bytes 0–29 des neuen Sprites (das später auf Y-Position 121 dargestellt wird [Spritepointer 16]) und zusätzlich die Bytes 30–62 des alten Sprites (das jetzt gerade auf Y-Position 100 dargestellt wird [Spritepointer 0]).
   Ich arbeite also mit „Zwischensprites“. Dieses Zwischensprite könnte z. B. im Spritepointer 8 gespeichert sein.

3. Bei Y-Position 121 muss ich nun nichts mehr machen. Ich habe durch diese „Zwischensprites“ Rechenzeit, um z. B. die X-Position oder Farben zu verändern, wenn das notwendig is

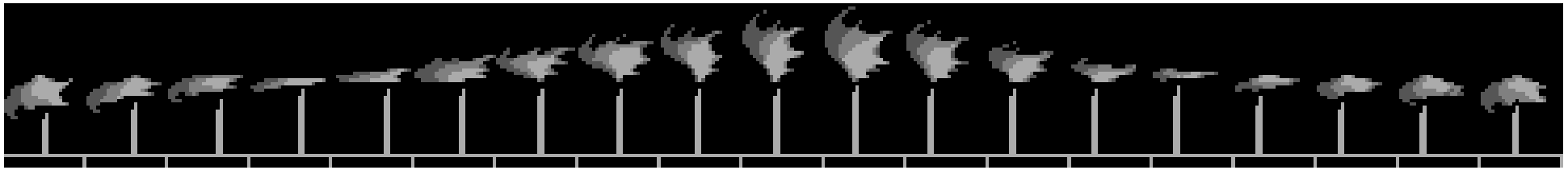
4. Ab Y-Position 122 (gemeint ist, wenn der Rasterstrahl die zweite Zeile des neuen Sprites darstellt) muss ich das vollständige Sprite einschalten, denn die Matrix des „Zwischensprites“ bildet ja nur bis Spritezeile 11 ab.

Für die Registeränderung bei Punkt 1, 2 und 4 habe ich viel Zeit – nämlich bis zu 9 Rasterzeilen.

Fliegender Spritedrache
Der Drachenkörper besteht nur aus drei animierten Multicolor-Sprites.
Die Flügel bestehen aus drei überlagerten Single-Color-Sprites, um hochauflösendes Multicolor zu suggerieren.
Insgesamt werden für den Flügel 57 Sprites für 19 Animationsschritte verwendet. Der Flügelschlag eines Weißkopfseeadlers war hier das Vorbild.

Meine Tochter Elisa hat ein Drachenflügel-Modell aus Pappe konzipiert, das mit Lötzinndraht beweglich gemacht wurde.
Mit einem Tablet wurde der Flügelschlag mit Mitwirkung meiner Tochter Julia als Stop-Motion aufgezeichnet – allerdings nur mit 11 Animationsschritten.

Die Fotos wurden auf C64-Graustufen angepasst, mit Pixcen überarbeitet und Zwischenschritte erstellt, um die Animation flüssiger zu gestalten.
Um möglichst die ganze Fläche der 24×21-Pixel-Matrix der Sprites zu nutzen, wurde während der Animation die Spriteposition zudem in X- und Y-Richtung angepasst (Ausgleichtabelle). Der Stab, an dem der Pappflügel befestigt war, wurde als Referenz für die X- und Y-Positionierung der Sprites verwendet.



(zoom in)

Sternenhimmel

Der Sternenhimmel im Upper Border besteht nur aus einem Sprite, das alle zwei Zeilen auf eine andere X-Position gesplittet wird.
Darunter werden die Sterne im Bitmap abgebildet, das ja für den Farbübergang schon auf Hires gesetzt ist.

Insgesamt mussten für diesen Part zwei VIC-Bänke verwendet werden, da der Speicherbedarf der Sprites zu hoch war.

Vielen Dank an Elisa, Julia, Carmen! und natürlich Abyss-Connection!

Abyss-Connection: Living for the Cursor