BeamConstruct – Linux in der Laserindustrie

von Hans Müller

Das bereits in vorangegangenen Artikeln vorgestellte OpenAPC-Softwarepaket (siehe „Heimautomatisierung für Hardwarebastler Teil 1–3“, freiesMagazin 01/2011 [1], 03/2011 [2] und 06/2011 [3]) hat mit der vor kurzem neu veröffentlichten Version 2 – welche auf den klangvollen Codenamen Aurora Borealis hört – einige umfassend neue Funktionalitäten erhalten, die einen etwas genaueren Blick auf diese Veränderungen rechtfertigen. Neben verschiedenen kleinen Detailverbesserungen, neuen Plug-Ins, die zusätzliche Hardware unterstützen und einer kleineren Umorganisation des gesamten Paketes sticht eine Änderung deutlich heraus: mit der Software *BeamConstruct* ist jetzt eine Applikation verfügbar, welche auf die Ansteuerung von Laserscannersystemen [4] und generell auf Lasermarkieroperationen hin optimiert ist.

Laserscanner?

Der Name indes ist für den unbedarften Benutzer etwas irritierend. Ist ein Scanner normalerweise ein Gerät, mit dem sich Dinge aus der realen Welt in digitale Informationen wie z. B. ein Bild oder ein 3D-Modell umwandeln lassen, so bezeichnet er hier etwas, was eine Oberfläche mit einem Laser abtastet. Das heißt, am Ende einer Bearbeitung mit einem solchen Laserscannersystem steht keine Datei und keine Datensammlung, vielmehr wurde das abgetastete Werkstück selbst verändert. Abhängig von der gewählten Laserleistung, der Geschwindigkeit des Bearbeitungsprozesses und des bearbeiteten Materials können dabei die unterschiedlichsten Aufgaben gelöst und Veränderungen am Material bewirkt werden. Das beginnt beim Abtragen von Oberflächen (z. B. um ein Material zu reinigen), geht weiter bei Lasergravierarbeiten (z. B. zum Beschriften und zur Fertigung von Schildern) sowie beim Laserschweißen und hört beim Laserschneiden noch lange nicht auf.

Solcherlei mit Lasern bearbeiteten Materialien sind dabei mittlerweile so weit verbreitet, dass man es manchmal kaum glauben mag, wofür diese Technik überall verwendet wird. Da ist beispielsweise die Handytastatur, deren Zahlen mit einem Laser erzeugt wurden, das Verfallsdatum oder der Gewinnspielcode auf Lebensmittelverpackungen oder Getränkeflaschen, die matt glänzende, „gebürstete“ Metalloberfläche hochwertiger Geräte oder aber auch die Grillstreifen auf manchen Fertig-Lebensmitteln.

Aber auch für andere Produktionsprozesse kommen Laser zum Einsatz, so beispielsweise bei der Herstellung von Solarzellen.

Die dafür eingesetzten Laserscannersysteme besteht dabei aus einem Scankopf [5], welcher zwei Spiegel beinhaltet, die den Laser in X- und Y-Richtung ablenken, dem Laser selbst und einer Controller-Elektronik, die Scankopf, Laser und die ansteuernde Software miteinander verbindet. Optional kann auch noch ein weiteren Element in der Strahlführung des Lasers vorkommen, welches die Position des Laserfokus verändert. Das ergibt die Möglichkeit, in Z-Richtung zu positionieren. Anwendungsgebiete hierfür sind das Rapid Prototyping [6] oder das In-Glass-Marking, bei dem dreidimensionale Objekte in einem Glasquader als solche sichtbar werden.

Linux!

Und genau an dieser Stelle kommt die jetzt neu im OpenAPC-Paket enthaltene Software *BeamConstruct* zum Einsatz. War es bisher so, dass einzelne Hersteller von Scannercontrollern Treiber für Linux anboten (z. B. Scanlab [7] für alle Karten ab der RTC3) oder aber generell ein plattformunabhängig zu benutzendes TCP/IP-Interface in diese implementiert haben (z. B. Raylase [8] in der SP-ICE2), so sah es bei der Anwendersoftware bisher schlecht aus. Lasermarkierapplikationen gab es nur für Windows; wer Linux benutzen wollte, konnte sich nur selbst etwas programmieren. Das ändert sich mit *BeamConstruct* jetzt grundlegend, hier ist erstmalig eine vollständige und umfassende, für Laserscannersysteme optimierte CAD-Applikation verfügbar, welche auf das wxWidgets-Toolkit [9] aufsetzt und deswegen für Windows und Linux zu haben ist. Unter Linux werden dann natürlich nur die Scannercontroller unterstützt, welche von Haus aus bereits Linux-Unterstützung bieten, allerdings ist das in dem Fall keine echte Ein-

schränkung: der Marktführer Scanlab ist bei diesen dabei und wenn sich die Software entsprechend ihres Potenzials durchsetzt, werden die anderen Hersteller früher oder später zwangsläufig folgen müssen.

Erste Schritte

Nach dem ersten Start von *BeamConstruct* zeigt sich dem Benutzer eine Oberfläche, die dem ebenfalls im OpenAPC-Paket enthaltenen CNConstruct sehr ähnlich ist. Tatsächlich scheinen beide Applikationen eine gemeinsame Basis zu nutzen. Das bietet sich auch an, da beide dem Benutzer ähnliche CAD-Funktionalitäten bieten, welche benötigt werden, um Prozessdaten zu erzeugen. Die Unterschiede beginnen bei deren Nutzung: Können die CNConstruct-Daten verwendet werden, um beispielsweise einen XY-Tisch und eine Fräse anzusteuern, so werden sie bei *BeamConstruct* eben zur Ansteuerung eines Scankopfes und eines Lasers verwendet. Grundsätzlich gilt die folgende Beschreibung aber in nahezu identischer Weise auch für CNConstruct.

Wer ein Laserscannersystem sein Eigen nennt und dieses mit *BeamConstruct* nutzen möchte, sollte die Applikation zuerst entsprechend der eigenen Hardware konfigurieren. Dazu findet sich im Menü „*Project*“ ein Untermenü „*Project Settings ...*“. Dieses öffnet das Fenster für globale Konfigurationen, im Panel „*Hardware*“ kann dann festgelegt werden, welche Geräte von der Software angesteuert werden können. So ist es zum einen möglich, aus einer Liste verfügbarer Treiber für Scannercontroller den passenden Typ auszuwählen und diesen anschließend mit einem Klick auf den Button „*Configure*“ entsprechend den eigenen Wünschen anzupassen. Nach dem gleichen Prinzip können bis zu zwei Motoren ausgewählt und eingestellt werden, mit denen es möglich ist, zusätzliche Bewegungen auszuführen. Damit lassen sich später komplexe Abläufe und Steuerungen realisieren wie beispielsweise das Bearbeiten eines Ringes, welcher dann Schritt für Schritt weiter gedreht wird, das Bearbeiten übergroßer Flächen, welche per zusätzlichem XY-Tisch unter dem Scanner weiter bewegt werden und vieles andere mehr.

An dieser Stelle sollte eine Warnung ausgesprochen werden: Die Laserbearbeitung ist ein gefährlicher Spaß. Kann ein unbedarfter Amateur mit einem vergleichsweise harmlosen Werkzeug wie einer Bohrmaschine beim unsachgemäßen Umgang schon für nennenswerte Schäden und ernsthafte Verletzungen sorgen, so sind Laser um ein vielfaches gefährlicher! Bereits Laserleistungen nur wenig oberhalb von 1 mW (entsprechend Laserklassen ab 3 [10]) können zum sofortigen Erblinden führen, bei entsprechend höheren Leistungen sind auch schwere Verbrennungen und Verletzungen möglich. Sicherheit bietet hier noch nicht einmal die Verwendung von handelsüblichen Laserpointern, da viele Billigprodukte ebenfalls eine viel zu hohe und hierzulande damit eigentlich nicht zulässige Leistung abgeben. Beim Experimentieren mit dieser Technologie ist auf jeden Fall äußerste Vorsicht geboten, die Einhaltung der Sicherheitsbestimmungen und die Einrichtung eines umfassenden Laserschutzes sind also unabdingbar!

Auch ist es keine Lösung, sich einfach nur vom Strahlweg des Lasers fern zu halten. Spätestens wenn dieser auf ein Objekt trifft, wird dessen Strahlung auch reflektiert. Hat das Objekt dann keine exakt ebene Oberfläche (wie es nur bei einem Spiegel der Fall wäre) ist die Reflexionsrichtung und Streuung der Laserstrahlung nicht vorhersehbar. Und auch wenn das Internet voll von Videos ist, in denen Leute scheinbar ohne Folgen völlig planlos mit Lasern herumhantieren, so ist das kein Grund für eine Entwarnung: wenn diese blind sind, sind sie sicher kaum noch in der Lage ein Video über diesen Zustand zu drehen und es bei YouTube hochzuladen.

Sind alle Maßnahmen getroffen, welche dafür sorgen, dass der Laser keinen Schaden anrichten kann, so ist auch die weitere Verwendung von *BeamConstruct* sorgenfrei möglich. Wurden alle Einstellungen vorgenommen, kann das erste Laserprojekt erzeugt werden.

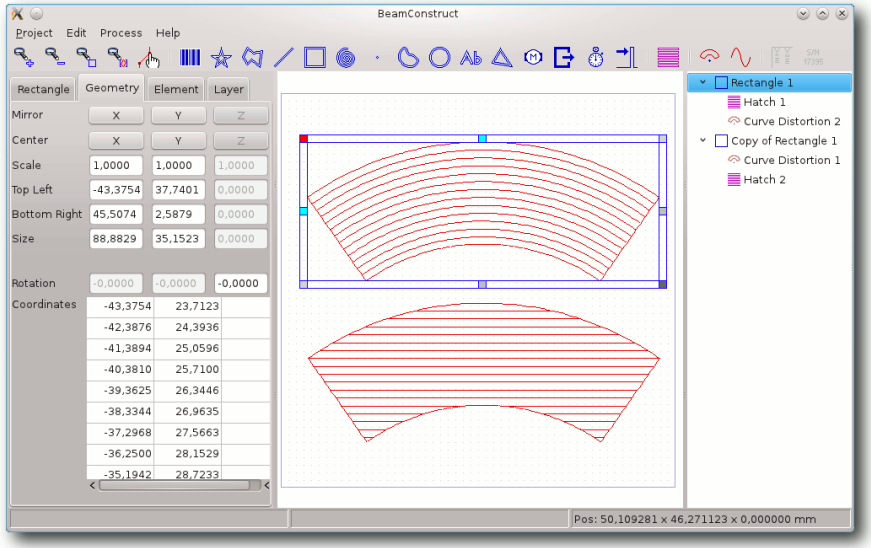
Geometrien und Hierarchien

So finden sich in der Werkzeugleiste am oberen Fensterrand verschiedene geometrische Grundformen wie z. B. ein Kreis, ein Rechteck, ein Dreieck aber auch ein Symbol für Text, für Barcodes und anderes mehr. Wird eines dieser Toolbar-Elemente selektiert, so kann es anschließend im Zeichenbereich erzeugt werden. Am Beispiel des Kreises sieht das so aus, dass ein erster kurzer Mausklick in den Zeichenbereich (die Maustas-

te dabei nicht festhalten!) die Position des Mittelpunktes festlegt. Wird die Maus dann weiter bewegt, zieht das den Kreis auf – der Radius ergibt sich aus dem Abstand zwischen der Position des ersten Mausklicks und der aktuellen Position des Mauszeigers. Ein weiterer Klick mit der linken Maustaste legt diesen Radius dann fest und beendet den Zeichenvorgang – der Kreis ist jetzt das erste Element in diesem Beispielprojekt.

Dieser Vorgang hat verschiedene Änderungen innerhalb der Oberfläche von *BeamConstruct* bewirkt: so wurde das neue Element „*Circle*“ rechts in eine Liste eingetragen, welche eine Übersicht über die vorhandenen Elemente eines Projektes bietet. Auf der linken Fensterseite wurde ein Panel „*Circle*“ zu den dort bereits verfügbaren Tabs hinzugefügt. Dieses Panel existiert immer nur so lange, wie ein Element innerhalb des Projektes ausgewählt ist. In diesem lassen sich die Eigenschaften eines solchen Elementes verändern. Im Falle eines Kreises sind das beispielsweise der Linientyp (durchgängig, gestückelt, gestrichelt, Punktmuster, . . .), dessen Anfangs- und Endwinkel sowie einige Parameter mehr, welche zum Teil nur für 3-D-Lasersysteme relevant sind, weil sie den Kreis um die dritte Dimension erweitern.

Eine wirklich interessante Änderung ergibt sich, wenn jetzt in der Toolbar der Button „*Hatch*“ (symbolisiert durch mehrere horizontale Linien in violetter Farbe) betätigt wird. Dieser fügt dem Kreis ein Füllmuster hinzu. Bereits bekannt: das Panel auf der linken Bildschirmseite zeigt jetzt die zu diesem Füllmuster gehörenden Parameter an; hier kann man beispielsweise den Winkel, die Art der Schraffierung, den Abstand von der umrahmenden Geometrie und anderes mehr einstellen. Neu ist die Art, wie dieses neue Element in der Liste auf der rechten Seite angezeigt wird: Jetzt offenbart sich diese als Baumdarstellung, innerhalb derer die Elemente eines Projektes hierarchisch angeordnet sind. So sind Hatch-Patterns



Die Reihenfolge macht's: Gleiche Zusatzelemente mit unterschiedlicher Wirkung, hier im Beispiel eines gehatchten Rechteckes.

(also die Füllmuster) immer einem grafischen Basiselement untergeordnet, da sie ja nur auf bereits vorhandene Geometrien aufbauen können.

Dieses Funktionsprinzip geht aber noch weiter, da es neben dem Hatch auch Elemente zur Nachbearbeitung von Geometrien gibt (in der Toolbar in roter Farbe dargestellt). Diese verändern die vorhandenen Geometrien entsprechend ihrer Eigenschaften und ihrer Position unterhalb eines solchen Basiselements. Soll heißen: eine Liste von Unterelementen wird immer genau in der vorgegebenen Reihenfolge abgearbeitet.

Das sieht auf den ersten Blick etwas kompliziert aus und kann bei umfangreichen Kombinationen von Elementen auch mal zu Verwirrungen führen, bietet tatsächlich aber deutliche Vorteile. So ist es nicht nötig, jedes grafische Element mit Unmengen von Parametern zu überladen, nur um auch noch den letzten geometrischen Kniff aus ihnen herauszuholen. Vielmehr entsteht mit dieser Methode eine Vielfalt von Möglichkeiten allein dadurch, dass Elemente frei kombinierbar sind.

Ein Beispiel ist ein Text, welcher gebogen werden soll, sodass er beispielsweise kreisförmig um etwas herum gelegt werden kann. Hier ist nichts weiter als ein Text-Basisobjekt (blaues Symbol „*Ab*“ in der Toolbar) sowie das „*Curve Distortion*“ Nachbearbeitungsobjekt (rotes Kreissegment-Symbol in der Toolbar) erforderlich. Das erste erzeugt den Text, das zweite – dem Text-Element als Unterobjekt hinzugefügt – verbiegt diesen Text.

Variationen und Kombinationen

An diesem Beispiel des radialen Textes lässt sich auch leicht das Prinzip der Reihenfolge der Unterelemente demonstrieren. Dazu soll in einem ersten Schritt wieder ein Text-Objekt erzeugt werden. Dieses erwartet nur einen einzelnen Mausklick im Zeichenbereich, um dessen Position festzulegen. Dieses Text-Element erhält als erstes Unterelement wieder ein Hatch-Pattern. Hier wird in den Hatch-Parametern lediglich der Linienstil von „*Continuous*“ nach „*Connected Lines*“ geändert. Ist das geschehen, sollte wieder das Textelement selbst selektiert werden, damit im nächsten Schritt das Curve-Nachbearbeitungselement als weiteres Unterobjekt hinzugefügt werden kann.

Jetzt ist zu sehen, dass das Curve-Element alle Geometrien um diesen Radius verbiegt, die bis zu diesem Zeitpunkt existieren, es wird also sowohl der Umriss des Textes als auch sein Füllmuster gebogen. Anders sieht es jedoch aus, wenn anschließend ein weiteres Hatch-Element hinzugefügt wird: Dessen Fülllinien werden durch das Curve-Element nicht beeinflusst, da es in der Reihenfolge der Unterelemente erst nach diesem kommt. Die zweiten Hatch-Linien sind gerade; ungeachtet der zuvor erfolgten Bearbeitung der Geometrien baut dieses Element auf den an dieser Stelle vorhandenen Daten auf.

Dieses Beispiel zeigt deutlich: allein durch die Kombination verschiedener Elemente und der Reihenfolge, in der diese gesetzt werden, sind die unterschiedlichsten Ergebnisse möglich – und das ohne sich mit komplizierten und verwirrenden Parametern herumschlagen zu müssen. Neben dieser logischen Struktur bringen zwar fast alle Elemente noch eigene Parameter mit, diese sind aber recht einfach und übersichtlich, da sie immer nur auf den Kontext des jeweiligen Elementes begrenzt sind und keine „Super-Geometrie“ existiert, die alles können muss.

Die Basiselemente

Diese Elemente – welche in der Applikation selbst als „Primary Element“ bezeichnet werden – werden in der Toolbar am oberen Bildschirm in blauer Farbe dargestellt. Die meisten von ihnen definieren grundlegende Geometrien, einige Sonderfälle erfüllen aber Steuerungsaufgaben und manifestieren sich deswegen auch nicht in Form einer neuen Grafik.

nicht erweiterbare Basiselemente	
Motor	fügt zusätzliche Bewegungsinformationen hinzu, welche bei der Abarbeitung des Projektes den konfigurierten Motor steuert.
Laseroutput	viele Scannercontroller besitzen zusätzliche digitale und analoge Ausgänge; diese können hiermit explizit auf bestimmte Werte gesetzt werden bzw. es ist möglich, Signale in Form von Pulsen auszugeben
Delay	stoppt die Abarbeitung eines Projektes für eine einstellbare Zeitspanne
External Trigger	wartet mit der Abarbeitung aller nach diesem Element folgenden Objekte, bis ein externes Triggersignal an der Scannerkarte festgestellt wurde

Basiselemente	
Dot	ein einzelner Punkt
Line	eine Linie
Circle	ein Kreis (der sich im 3D-Modus auch in Form einer Spiralfeder in die Tiefe fortsetzen kann)
Spiral	eine Spirale (die im 3D-Modus auch in die Tiefe gehen kann)
Rectangle	ein Rechteck
Triangle	ein Dreieck
Star	ein sternförmiges, rotationssymmetrisches Objekt
Polygon	ein frei definierbares, offenes oder geschlossenes Vieleck
Bezier Curve	eine Abwandlung eines Polygons, bei dem die Eckpunkte nicht direkt miteinander verbunden sind sondern die Stützpunkte einer Bezier-Kurve definieren [11]
Text	ein Element zur Textdarstellung; Zeichensatz, Schriftart, Zeichenabstand und vieles mehr sind über die Parameter wählbar
Barcode	erzeugt einen Barcode aus vorgegebenen Daten; Typ, Encoding, Darstellung und anderes mehr sind per Parameter wählbar wobei hier alle wichtigen Barcodetypen wie QR, EAN, DataMatrix sowie diverse Pharma-Barcodes unterstützt werden.

Die Symbole der Basiselemente in der Toolbar.

Basiselemente, welche keine neue Geometrie erzeugen, können logischerweise auch nicht mit Unterelementen um Geometrien erweitert bzw. modifiziert werden.

Das Zusatzelement „Hatch“

Innerhalb der Software existiert zwar ein eigener Elementtyp „Additional Geometry“ (in der Toolbar durch ein violettes Symbol dargestellt), allerdings gibt es derzeit tatsächlich nur ein Element, welches zu dieser Gruppe gehört: der eingangs bereits erwähnte Hatcher. Dieser erlaubt es, bestehende Geometrien zu füllen, um so als Ergebnis einer Laserbearbeitung nicht nur die Kante eines Objektes zu sehen, sondern auch dessen Fläche. Hier kommt der Unterschied zwischen Laserschneiden und -gravieren zum Tragen: würde beim Schneiden die Kante genügen, um ein Objekt mit einem bestimmten Umriss auszuschneiden, so ist beim Lasergravieren die Veränderung der Fläche interessant: Ein beispielsweise auf ein Objekt gelaserter Text soll auch ausgefüllt werden, so dass nicht nur dessen Umriss als Veränderung im Material sichtbar wird, sondern auch die Flächen der Buchstaben.

Das einzige Zusatzelement der Werkzeugleiste ist der Hatcher.

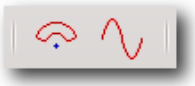
Die Nachbearbeitungselemente

Damit die Wirkung dieser Elemente mit dem offiziellen Namen „Postprocessing Element“ sich auch in sichtbaren und vor allem in den gewollten Änderungen niederschlägt, ist es meistens erforderlich, für die vorangegangenen Basis- oder Zusatzelemente in deren Parametern einen anderen Linientyp als „*Continuous*“ zu wählen. Die Erklärung dafür ist denkbar einfach: Nachbearbeitungselemente können nur bereits existierende Daten verändern, sie fügen keine neuen Geometrien hinzu. Besteht eine lange Linie dann aber nur aus einem Anfangs- und einem Endpunkt, so kann das Element auch nur genau diese beiden Punkte verändern. Und damit lässt sich keine andere Form als eine Linie erzeugen.

Besteht die Kante eines Objektes jedoch aus vielen kurzen Strichen, so können die vielen Stützpunkte dieser Linie auch zu einer neuen Linienform verändert werden.

Sine Distortion – überlagert die Linien einer Geometrie mit einer Sinuswellenform, sowohl die Richtung der Ausbreitung als auch die Amplitude und Frequenz können dabei festgelegt werden.

Curve Distortion – biegt die übergeordneten Geometrien um einen Mittelpunkt herum, dessen relative Position, Ausrichtung, Orientierung und Biegeradius festgelegt werden kann.



Die Nachbearbeitungselemente.

Die Eingabeelemente

Dieser Elementtyp wird in der Toolbar in grün dargestellt und genießt eine Sonderrolle. So kann zu jedem Basiselement maximal ein Eingabeelement existieren. Das aber auch nur dann, wenn das Basiselement mit den Daten eines solchen Eingabeelements umgehen kann.

Die Eingabeelemente heben sich in der Toolbar durch grüne Symbole ab.

Das verdeutlicht sich vielleicht an einem Beispiel: Sowohl der Text als auch das Barcode-Basiselement besitzen in ihrem Konfigurationspanel Eingabefelder, in denen die Daten hinterlegt werden können, welche in Form von Text dargestellt oder in den Barcode hineinverschlüsselt werden können. Genau diese Daten können für diese beiden Elemente aus eben so einem Eingabeelement kommen. Das klingt unspektakulär, ist es im Fall des zweiten Eingabeelements aber definitiv nicht.

OAPC-Input

Wie später noch gezeigt wird, kann ein *BeamConstruct*-Projekt auch innerhalb eines OpenAPC-Visualisierungsprojektes verwendet werden. Dort existiert ein Plug-In, welches mit diesen Projektdaten umgehen kann und welches zusätzliche Dateneingänge besitzt. Genau einer diese Dateneingänge wird über die Parameter dieses Eingabeelements auf das übergeordnete Basiselement umgeleitet. Das heißt nichts

anderes, als dass der Inhalt eines *BeamConstruct*-Projektes auch außerhalb von *BeamConstruct* in einer Maschinensteuerung verändert werden kann, was sich beispielsweise nutzen lässt, um ständig wechselnde Texte auf verschiedene Werkstücke zu bringen

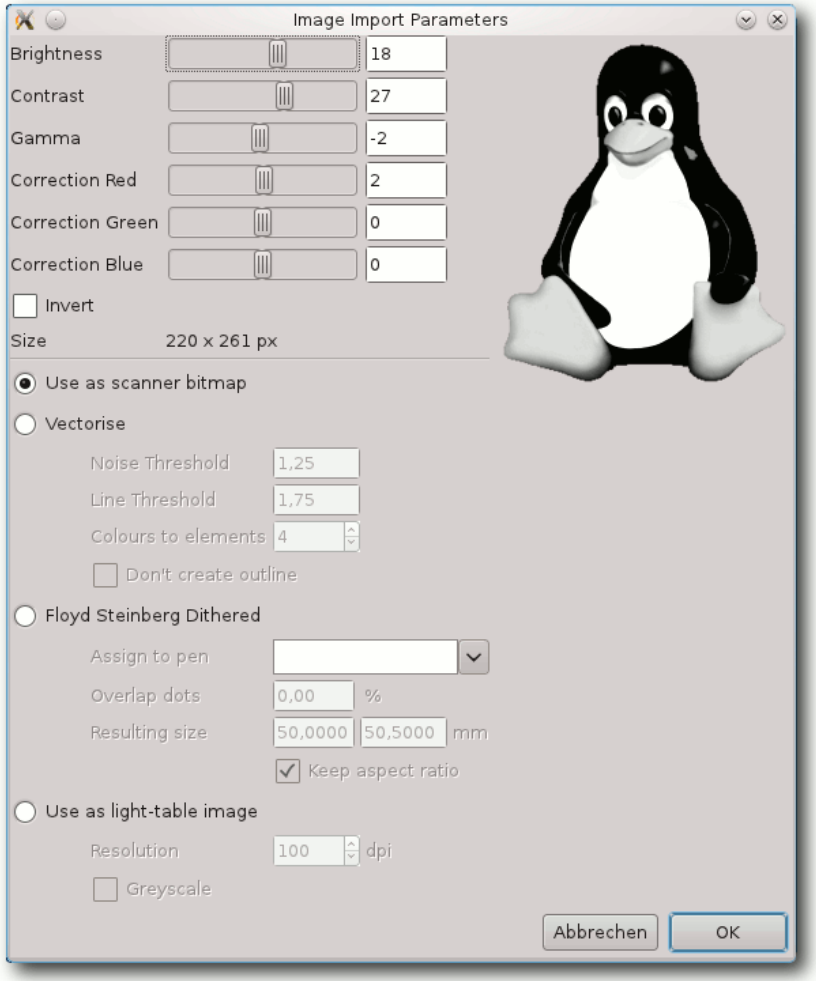
Serial/Date/Time

Der Name dieses Elements deutet bereits an, was damit möglich ist: es lassen sich sowohl Informationen zu Datum und Uhrzeit als auch Seriennummerndaten generieren, welche dann über den Umweg des zugehörigen Basisobjektes auf das zu bearbeitende Material aufgebracht werden können. Dabei sind Kombinationen aus frei wählbarem Text, sich bei jeder Markieroperation verändernden Zählern, Datums- und Zeitinformationen in unterschiedlichster Formatierung möglich, welche es erlauben, die vielfältigsten Anwendungsgebiete abzudecken. So ist es mittels dieses eher unscheinbaren Elementes möglich, so unterschiedliche Dinge wie Produktionsdaten, Schichtinformationen, Verfallsdaten, Seriennummern und vieles andere mehr zu erzeugen – um die Aktualisierung kümmert sich die Applikation dabei vor jedem Markierprozess selbst.

Import von Rastergrafiken

Neben der Möglichkeit, eigene Projekte zu erstellen und Vektorgrafiken diverser Formate zu importieren, können auch sogenannte Pixel- oder Rastergrafiken importiert werden. Das sind nichts anderes als die allseits bekannten Bildformate, welche als JPEG, PNG, GIF, BMP oder anderes mehr daher kommen können.

Abhängig vom verwendeten Lasertyp können damit dann reine schwarz-weiß-Grafiken (z. B. mit CO2-Lasern) als auch echte Graustufen-Bilder (z. B. mit YAG-Lasern [12]) auf ein Werkstück aufgebracht werden.



Der Import-Dialog für Raster-Images, Scanner-Bitmaps können auch anschließend noch weiter verändert werden.

Wurde ein entsprechendes Bild über den Menüpunkt „*Project* → *Import*“ geladen, öffnet sich ein Dialog, in dem grundlegende Bildparameter angepasst werden können. Neben einfachen Möglichkeiten, das Bild noch ein wenig zu verändern, kann hier auch festgelegt werden, wie es innerhalb eines *BeamConstruct*-Projektes eingesetzt werden soll.

Use as Scannerbitmap

Wird diese Option gewählt, so wird das Bild auf eine Art in das Projekt importiert, die dafür sorgt, dass es durch den Laser direkt als echte Bitmap ausgegeben wird. Dabei entscheidet der zuvor für die Scannerkarte gewählte Lasertyp, ob das Bild in echten Graustufen oder als Schwarzweißbild wahlweise mit Floyd-Steinberg-Dithering dargestellt wird

Vectorise

Das Bild wird vektorisiert, d. h. die Pixeldaten werden in Vektorlinien umgewandelt. Dieser Vorgang ist recht kompliziert und das Ergebnis hängt auch sehr stark von den gewählten Vektorisierungsparametern sowie der Qualität des geladenen Bildes ab. Generell gilt: ein starker Kontrast, eine hohe Auflösung, wenig Rauschen und erst recht keine Kompressionsartefakte

sind wichtige Voraussetzungen für brauchbare Ergebnisse. Dennoch sollte man auch dann immer noch eine gewisse Zeit einplanen, um mit den vorhandenen Parametern zu spielen.

Floyd Steinberg dithered

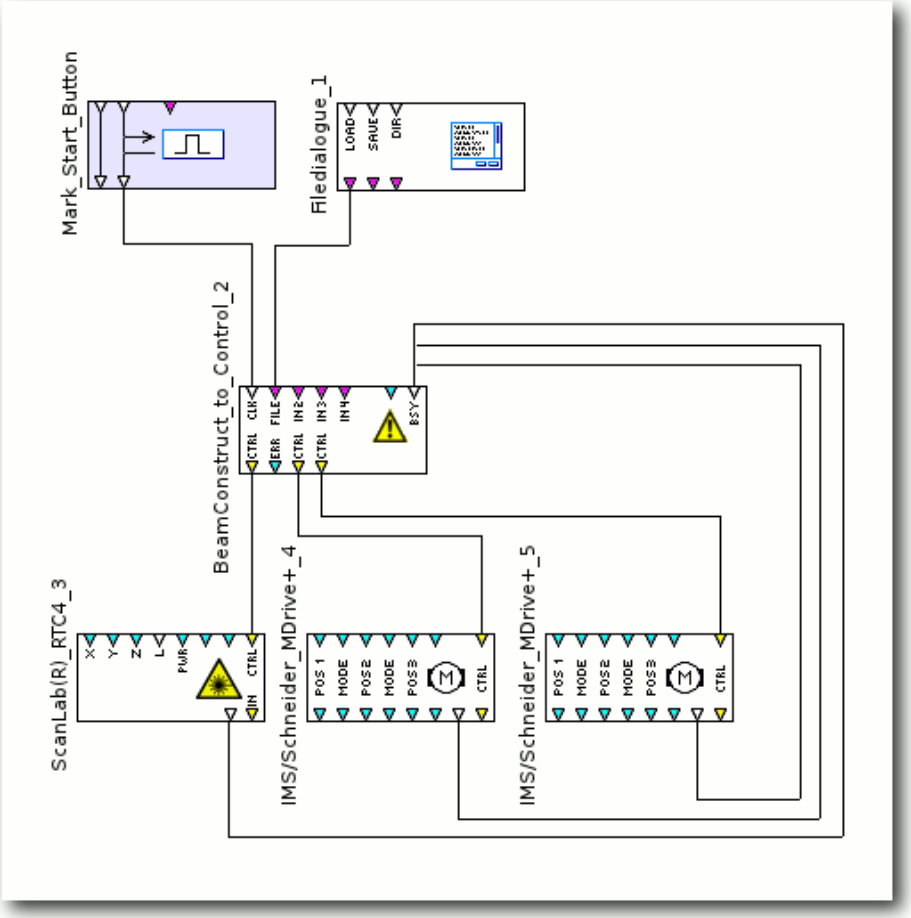
Auch diese Importmethode wandelt das Bild in Vektordaten um – dieses Mal allerdings ausschließlich in einzelne Punkte [13]. Die Anordnung dieser Punkte ist auf eine bestimmte Größe sowie die Spot-Größe des Lasers hin optimiert, sodass nach dem Import keinerlei Skalierfunktionen mehr auf das Ergebnis angewendet werden sollten.

Use as light-table image

Wird diese Option gewählt, so wird das Bild 1:1, ggf. sogar in Farbe und ohne Modifikation in das Projekt übernommen. Allerdings kann so ein Bild nicht über den Laser ausgegeben werden, vielmehr dient es als Vorlage, um eigene Grafiken zu zeichnen. Der Name „Light-table image“ – frei übersetzt also „Leuchttischvorlage“ – weist auch auf diesen Zweck hin; diese Bilder dienen als Vorlage für eigene Geometrien.

Integration in ControlRoom-Projekte

Wurde *BeamConstruct* entsprechend der vorhandenen Hardware konfiguriert, so ist es über den Menüpunkt „*Process* → *Mark*“ möglich, einen Lasermarkierprozess direkt aus der Applikation heraus zu starten. Dabei werden alle vorhandenen Elemente angesteuert: sowohl der Laser als auch der Scankopf arbeiten entsprechend der Vektordaten und den für diese hinterlegten



Mögliche Verdrahtung eines ControlRoom-Projektes mit RTC4 und zwei MDrive-Motoren: alle BSY-Ausgänge müssen mit dem BSY-Eingang des Konverter-Plug-Ins verbunden sein.

Laser- und Scannerparameter (wie beispielsweise Markiergeschwindigkeit, Laserleistung und -frequenz) und die optional vorhandenen Motoren werden so verfahren, wie es im Projekt definiert wurde. Des Weiteren wird vor dem Start einer Laserausgabe ein Projekt auch immer vollständig aktualisiert: so werden dynamische Elemente wie Datum/Zeit oder Seriennummern auf den aktuellen Stand gebracht bzw. weitergezählt.

Da es allerdings die Ausnahme darstellen dürfte, dass die Applikation, mit der die Prozessdaten erstellt werden, auch die zugehörige Maschine ansteuert, gibt es auch noch einen weiteren Weg: Die Prozessvisualisierungs- und -steuerungssoftware „ControlRoom“, welche ebenfalls Teil des OpenAPC-Paketes ist, kann *BeamConstruct*-Projekte laden und abarbeiten. Das geschieht mit Hilfe des zugehörigen Plug-Ins „BeamConstruct to Control“ welches diese Projekte laden und in Control-Daten umwandeln kann. Diese werden dann über den Ausgang OUT0 an den Scannercontroller weiter gegeben. Die Ausgänge OUT2 und OUT3 sind optional, sollten aber mit ein oder zwei Motorcontrollern verbunden sein, sofern die geladenen Projekte diese Motoren erwarten. Existiert so ein Motor jedoch nicht, so wird ein *BeamConstruct*-Projekt, welches diesen

erwartet, bis in alle Ewigkeit auf die Fertigstellung dieser Bewegung warten.

Daraus ergibt sich auch eine weitere Notwendigkeit: Die BSY (=busy)-Ausgänge aller an das *BeamConstruct*-Plug-In angeschlossenen Komponenten müssen mit dessen BSY-Eingang verbunden werden. Über diese Leitung wird signalisiert, ob eine Operation – sei es nun ein Markiervorgang oder eine Motorbewegung – noch läuft oder ob diese fertiggestellt wurde und zum nächsten Bearbeitungsschritt übergegangen werden kann.

BeamConstruct versus CNConstruct

Wie eingangs erwähnt sind sich *BeamConstruct* und CNConstruct sehr ähnlich. Um genauer zu sein, leistet CNConstruct in weiten Teilen das Gleiche wie *BeamConstruct*, allerdings auf einer wesentlich generischeren Basis. Die dort erstellten Geometrien sind nicht bereits für eine bestimmte Art der Bearbeitung optimiert, vielmehr bleibt es hier völlig offen, wie der eigentliche Prozess aussieht.

Anders ist das bei *BeamConstruct*: hier sind verschiedene zusätzliche Funktionalitäten enthalten, welche nur für die Laserbearbeitung sinnvoll sind; auch sind die Penparameter bereits auf das Ausgabemedium „Laser“ abgestimmt. Ein weiterer deutlicher Unterschied: CNConstruct erlaubt nur eine Simulation des Prozesses, *BeamConstruct* kann vorhandene Hardware auch selbst ansteuern und so einen Prozess ablaufen lassen.

Sieht man hiervon einmal ab, sind sich beide aber durchaus ähnlich. So kann man die hier beschriebene Vorgehensweise für das Erstellen von Projekten, Hierarchien und Kombinationsmöglichkeiten von Elementen 1:1 für CNConstruct übernehmen – das grundlegende Bedienkonzept ist bei beiden Applikationen das Gleiche.

Licht versus Schatten

Zu jeder Programmvorstellung gehört immer auch eine gehörige Portion Kritik der Mängel (die jede Software hat), der Softwarefehler und sonstigen Unzulänglichkeiten. So etwas gibt es auch bei *BeamConstruct*; ab und zu findet sich mal ein unerklärlicher Absturz (weswegen es sich empfiehlt, ein Projekt öfter mal zu speichern), hin und wieder tut die Applikation nicht wirklich das, was man erwartet.

Allerdings fällt es schwer, das jetzt als echten Kritikpunkt hervorzuheben, da die hier getestete Version aus dem OpenAPC 2.0 Paket noch als Alpha-Version gekennzeichnet ist. Damit werden eigentlich Softwarestände gekennzeichnet, welche sich in der Regel irgendwo im Zustand „experimentell“ befinden. Das gilt in diesem Fall ganz klar nicht; mit der Software lässt sich arbeiten und sie ist definitiv benutzbar. Es bleibt also abzuwarten, welche Verbesserungen sich bis zur ersten stabilen Release noch ergeben.

Auch wenn aus diesem Grund ebenso mit Lob gespart werden soll, eins bleibt hervorzuheben: dass es mit *BeamConstruct* jetzt auch eine Lasermarkiersoftware für Linux gibt, bedeutet für die gesamte Laserbranche ein echtes Novum – und setzt für die Zukunft hoffentlich Maßstäbe!

LINKS

[1] http://www.freiesmagazin.de/freiesMagazin-2011-01

[2] http://www.freiesmagazin.de/freiesMagazin-2011-03

[3] http://www.freiesmagazin.de/freiesMagazin-2011-06

[4] http://de.wikipedia.org/wiki/Laserscanning#Materialbearbeitung_und_Fertigung

[5] http://de.wikipedia.org/wiki/Laserscanning#Scankopf

[6] http://de.wikipedia.org/wiki/Rapid_Prototyping

[7] http://www.scanlab.de

[8] http://www.raylase.com/

[9] http://www.wxwidgets.org/

[10] http://de.wikipedia.org/wiki/Laser#Laser-Klassen

[11] http://de.wikipedia.org/wiki/Bézierkurve

[12] http://de.wikipedia.org/wiki/Nd:YAG-Laser

[13] http://de.wikipedia.org/wiki/Floyd-Steinberg-Algorithmus

Autoreninformation

Hans Müller ist als Elektroniker beim Thema industrielle Automatisierung mehr der Hardwareimplementierung zugeneigt als dem Softwarepart und hat auch schon diverse Geräte in der privaten Wohnung verkabelt.

Diesen Artikel kommentieren