Fraktale
oder
Wie lang ist Englands Küste wirklich?

Entwickler-Handbuch

Inhaltsverzeichnis

Kompilierung

Leider konnten wir bis jetzt kein mit den GNU autotools erstelltes configure-Skript und Makefile bereitstellen. Daher kann es sein das die mitgelieferten Makefiles nicht auf allen Systemen funktionieren. Um die Kompilierung einzuleiten, muss zuerst `./fix_unix.sh` (unter Linux) oder `fix_mingw.bat` (unter Windows) ausgeführt werden. Anschließend wird das Programm durch Eingabe von `make` kompiliert und der Inhalt des Ordners „bin“ kann an den gewünschten Installationsort kopiert werden. (Das Programm muss Schreibrechte im Installationsordner haben.)

Weitere Targets der Makefile

data

Kopiert von FractalViewer und Hauptprogramm gemeinsam genutzte Daten in den „bin“ Ordner.

clean

Löscht die Oject-Dateien für eine erneute Kompilierung.

clean-all

Löscht die Object-Dateien und den gesamten Inhalt des „bin“-Ordners.

Erstellung und Bearbeitung von Textseiten

Dateistruktur

Der Ordner „data/fraktale/content/“ enthält in den Unterverzeichnissen die verschiedenen Kapitel mit den Textseiten. Pro Textseite gibt es ein JPG-Bild mit der Nummer der Textseite (0.jpg ist die erste) und einer Auflösung von 800 mal 600 Pixeln. Dieses Bild enthält alles, was auf der entsprechenden Seite dargestellt werden soll. Weiterhin kann es zu einer Textseite eine Datei mit der Endung „.gui“ geben (z.B. 0.gui). In dieser Datei stehen Informationen zu zusätzlichen Elementen für die GUI auf dieser Seite. Ist zu einer Textseite eine mp3-Datei (z.B. 0.mp3) vorhanden, so wird automatisch ein Wiedergabe-Button für diese Sounddatei angezeigt.
Sollte eine Ordner eines Kapitels einen Ordner namens „moreinfo“ enthalten, wird am Ende des Kapitels automatisch der Button für weiterführende Informationen angezeigt.

Erstellen von Textseiten

Zur Erstellung von Textseiten kann jedes Programm verwendet werden, dass JPGs speichern kann. (Natürlich gehen auch andere Grafikformate, wenn man diese zu JPGs konvertiert). Wir empfehlen das frei verfügbare OpenOffice (Impress). So sind auch alle Textseiten (bei den Sourcecode-Versionen) nochmals in diesem Format gespeichert. Um neue Textseiten zu erstellen kann die Vorlage „template.sti“ im Ordner „data/fraktale/openoffice/“ verwendet werden. Diese enthält ein Makro, dass die Präsentation in BMP-Bilder exportiert. Diese können dann mit dem Linux-Shell-Skript „gfx_conv.sh“ in JPG-Bilder umgewandelt werden. Dabei ist das erste Argument das Verzeichnis, in dem sich die Bilder befinden. Das OpenOffice-Makro exportiert nicht gleich in JPG-Bilder, um eine bessere Qualität zu erreichen.

GUI-Dateien

Bei den GUI-Dateien handelt es sich um XML-Dateien. Allerdings wird keine DTD und kein <xml ...>-Tag angegeben. Auf Grund eines Bugs muss bei Kommazahlen ein Komma und kein Punkt (wie in der Programmierung normalerweise) verwendet werden (nicht 12.34 sonder 12.34). Folgende Tags sind möglich:

gui
Sämtliche anderen Tags müssen von diesem Tag eingeschlossen sein.

qcheck
Stellt eine Radio-Button oder eine Checkbox dar.
x, *y*: Gibt die Position des Elements auf dem Bildschirm an.
label: Gibt die Beschriftung an
type: radio oder check
rightanswer: Gibt an, ob dieses Element für eine richtige oder falsche Antwort steht. Gültige Werte sind yes und no. Bei nicht Angabe wird no angenommen.
group: Gibt die Gruppe an zu der das Element gehört. Eigentlich nur für Radio-Buttons entscheident.

qedit

Stellt eine Eingabefeld dar.
x, *y*: Gibt die Position des Elements auf dem Bildschirm an.
label: Gibt die Beschriftung an
maxchar: Gibt die maximale Anzahl an Zeichen an, die eingegeben werden kann.
rightanswer: Gibt an, welche Eingabe als richtig gewertet werden soll.
type: Gibt an, ob ein kleines oder großes Eingabefeld dargestellt werden soll. Gültige Werte sind small und large.

xma

Dient zur Wiedergabe einer XMA-Datei.
x, *y*: Gibt die Position des Elements auf dem Bildschirm an.
src: Gibt die Datei an, die wiedergegeben werden soll.
autostart: Gibt an, ob die Animation automatisch abgespielt werden soll. Gültige Werte sind yes und no.
loop: Gibt an, ob die Animation in einer Schleife abgespielt werden soll. Gültige Werte sind yes und no.

xmacontrol

Dient zur Anzeige eines XMA-Steuerelemts auf dem Bildschirm.
x, *y*: Gibt die Position des Elements auf dem Bildschirm an.
type: Gültige Werte sind: start, stop, pause.

iter

Dient zur Anzeige eines interaktiven Elements für grafische Iteration.
x, *y*: Gibt die Position des Elements auf dem Bildschirm an.
xmin, *xmax*, *ymin*, *ymax*: Geben den Bereich des Koordinatensystems an.
xres, *yres*: Gibt die Auflösung an und bestimm somit die Breite des Elements.
my: Gibt den Wert der Variablen μ in der logistischen Funktion an.
n: Gibt die Anzahl der jeweils durchzuführenden Iterationsschritte an.
lower, *greater*: Geben die untere und obere Grenze für gültige Eingaben an. Können weggelassen werden.

iteredt

Dient zur Anzeige eines Eingabefeldes für grafische Iteration. Nur in Verbindung mit dem iter-Tag.
x, *y*: Gibt die Position des Elements auf dem Bildschirm an.
label: Gibt die Beschriftung an
maxchar: Gibt die maximale Anzahl an Zeichen an, die eingegeben werden kann.
type: Gibt an, ob ein kleines oder großes Eingabefeld dargestellt werden soll. Gültige Werte sind small und large.

iterbut

Dient zur Anzeige eines Buttons, der eine grafische Iteration startet. Nur in Verbindung mit dem iter-Tag und dem iteredt-Tag.
x, *y*: Gibt die Position des Elements auf dem Bildschirm an.
label: Gibt die Beschriftung an

XMA-Dateien

Bei den XMA-Dateien handelt es sich wie bereits erwähnt, um ein spezielles Animationsformat. XMA steht für eXtensible Markup Animation und XMA-Dateien sind wie der Name vermuten lässt auch XML-Dateien. Auch bei diesen Dateien wird keine DTD und kein <xml ...>-Tag angegeben. Auch hier müssen Kommazahlen mit Komma statt Punkt angegeben werden.
Zeitwerte werden in Frames angegeben, wobei 1 Sekunde = 25 Frames gilt.
Bei der Angabe der Startzeitpunkte ist zu beachten, dass dieser relativ zum vorhergehenden Element ist, wenn die Elemente direkt dem root-Tag <animation> untergeordnet sind. Steht das Tag allerdings einem anderen Element untergeordnet (z.B. <fadein>), so bezieht sich der Startzeitpunkt auf den Startzeitpunkt des übergeordneten Elements. Wird kein Startwert angegeben, so wird null angenommen, wobei auch die eben erwähnten Punkte zu beachten sind. Negative Startwerte sind möglich.
Wird keine Anzeigelänge angegeben, so wird die längst mögliche genommen (außer beim animation-Tag). Gleiches gilt für Breite und Höhe (animation-Tag auch wieder ausgenommen). Die Position wird standardmäßig auf 0,0 gesetzt.
Folgende Tags sind möglich:

animation

Muss jede XMA-Datei umschließen.

length: Gibt die Gesamtlänge der Animation an.
w, *h*: Gibt die Breite und Höhe der Animation an.

group
Dient zur Gruppierung von Elemente, um viele Elemente mit gleichem Startzeitpunkt zu erstellen.
start: Gibt den Startzeitpunkt an. Siehe dazu auch die Einleitung zu diesem Kapitel.
length: Gibt Anzeigelänge der Gruppierung an.

fadein, **fadout**
Die von diesem Tag umschlossenen Elemente werden langsam ein- bzw. ausgeblendet.
start: Gibt den Startzeitpunkt an. Siehe dazu auch die Einleitung zu diesem Kapitel.
length: Gibt die Anzeigedauer an.
x, *y*: Gibt die Position des fadein-Bereichs an.
w, *h*: Gibt die Breite und Höhe des fadein-Bereichs an.

move
Bewegt die von diesem Tag eingeschlossenen Elemente.
start: Gibt den Startzeitpunkt an. Siehe dazu auch die Einleitung zu diesem Kapitel.
length: Gibt die Anzeigedauer an.
x, *y*: Gibt die Start-Position des move-Bereichs an.
w, *h*: Gibt die Breite und Höhe des move-Bereichs an.
endx, *endy*: Gibt die Endposition des move-Bereichs an.

fzoom
Zoomt ein Fraktal.
start: Gibt den Startzeitpunkt an. Siehe dazu auch die Einleitung zu diesem Kapitel.
length: Gibt die Anzeigedauer an.
x, *y*: Gibt die Position des fzoom-Bereichs an.
w, *h*: Gibt die Breite und Höhe des fzoom-Bereichs an.
src0, *src1*: Gibt die Quellbilder an (*src0* für das Ausgangsbild, *src1* für das Endbild)
data0, *data1*: Gibt die Koordinaten der Bilder an. Format: „xmin;xmax;xmin“ oder „xmin;xmax;ymin;ymax“

zoom
Zoomt die von diesem Tag eingeschlossenen Elemente.
start: Gibt den Startzeitpunkt an. Siehe dazu auch die Einleitung zu diesem Kapitel.
length: Gibt die Anzeigedauer an.
x, *y*: Gibt die Start-Position des zoom-Bereichs an.
w, *h*: Gibt die Start-Breite und -Höhe des zoom-Bereichs an.
endx, *endy*: Gibt die Endposition des zoom-Bereichs an.
endw, *endh*: Gibt die End-Breite und -Höhe des zoom-Bereichs an.

stretch
Streckt/staucht die von diesem Tag umschlossenen Elemente.
start: Gibt den Startzeitpunkt an. Siehe dazu auch die Einleitung zu diesem Kapitel.
length: Gibt die Anzeigedauer an.
x, *y*: Gibt die Position des stretch-Bereichs an.
w, *h*: Gibt die Breite und Höhe des stretch-Bereichs (die Fläche auf der untergeordnete Elemente zeichnen können) an.
neww, *newh*: Gibt die Breite und Höhe an, auf die gestreckt/gestaucht wird.

img
Zeigt ein Bild an.
start: Gibt den Startzeitpunkt an. Siehe dazu auch die Einleitung zu diesem Kapitel.
length: Gibt die Anzeigedauer an.
x, *y*: Gibt die Position des Bildes an.
src: Gibt die Bildquelle an.
trans: Gibt den Transparenzmodus an. Gültige Werte sind yes (RGB-Farbe 255, 0, 255 ist transparent), no (keine Transparenz), alpha (der Alpha-Kanal eines tga-Bildes wird verwendet). Dieses Attribut sollte auf jeden Fall einen sinnvollen Wert haben (bei tga-Bild mit Alpha-Kanal muss es alpha sein, sonst darf es nicht alpha sein), da das Bild sonst nicht angezeigt wird.
size: Gibt die Textgröße an.
color: Gibt die Textfarbe an. Format: „r;g;b“

Fragen und Antworten

Warum werden die Textseiten als Grafiken gespeichert?

Diese Frage ist durchaus berechtigt, denn es gibt einige Nachteile: Grafiken sind unflexibel und benötigen mehr Speicherplatz, als wenn man den reinen Text speichern würde. Allerdings fehlte uns die Zeit (da wir das Programm ja für einen Wettbewerb geschrieben haben und so zu einem bestimmten Zeitpunkt fertig sein musste) es so zu Programmieren, das die Texte wirklich als Text hätten gespeichert werden können, da dann automatisch die Stellen für Zeilenumbrüche bestimmt werden müssten und einen Interpreter für die Formeln benötigten. Natürlich hätte man dann auch nur die Formeln als Grafik speichern können. Aber dann bräuchte man erstmal einen Formeleditor, der als Grafik exportieren könnte, und einen solchen haben wir nicht gefunden.

Warum haben wir gettext bei FractalViewer, aber nicht beim Hauptprogramm eingesetzt?

Erst einmal die Gründe, warum wir beim Hauptprogramm kein gettext eingesetzt haben: Die Masse an Text im Hauptprogramm, die übersetzt werden müsste ist sehr groß. Den gesamten Text zweisprachig abzufassen war uns in der Zeitspanne für den Wettbewerb, für den wir das Programm geschrieben haben, einfach nicht möglich. Um Probleme zu vermeiden, wenn die Spracheinstellungen beim Benutzer nicht richtig gesetzt sind, haben wir uns entschlossen gettext vorerst nicht einzusetzen.