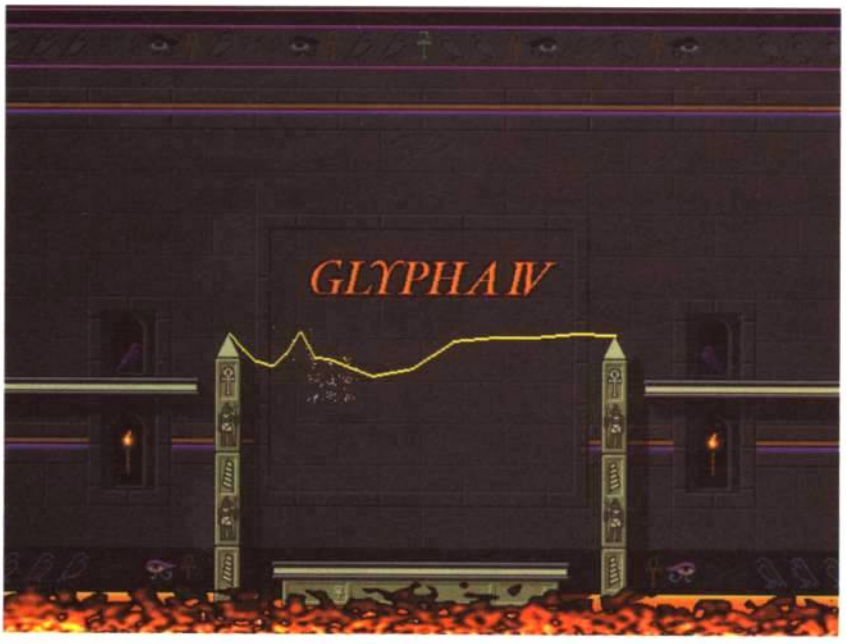


Game on!

Die Apple GameSprockets

Vor etwas über einem Jahr hat Apple – endlich – erkannt, daß es Leute gibt, die auch auf ihrem professionellen Computer spielen wollen. Um eine umfassende Unterstützung für Spieleprogrammierer zu bieten, wurden die „Game Sprockets" für PowerMacs entwickelt. Was es damit auf sich hat, wollen wir in diesem Monat klären.

Die Game Sprockets sind eine Sammlung an verschiedenen APIs, welche das Entwickeln von Spielen für das MacOS erheblich erleichtern. Die Sprockets teilen sich momentan in folgende Bereiche auf: DrawSprocket, InputSprocket, SoundSprocket, SpeechSprocket, NetSprocket und QuickDraw 3D RAVE. Alle Sprockets sind voneinander unabhängig. Man kann also in seinem Spiel z.B. nur die DrawSprocket nutzen, ohne den zusätzlichen Overhead der anderen APIs zu erzeugen. Wenden wir uns nun aber den einzelnen Sprockets zu:

DrawSprocket

Dieses API erlaubt es dem Programmierer, den für sein Spiel passenden Bildschirm in einer für ihn optimalen Grafikauflösung zu finden. Man erinnere sich: ein Mac kann mehrere angeschlossene Monitore haben, und jeder Monitor kann verschiedene Auflösungen darstellen können. DrawSprocket übernimmt automatisch das Wegblenden der Menüleiste, dem Control-Strip-Menü und anderen Fenstern und schaltet auf eine passende Auflösung für das Spiel – wenn der Monitor dies erlaubt. Vorbei sind die Zeiten, wo Spiele erforderten, daß der Benutzer selbst die Auflösung und Farbtiefe ändert!

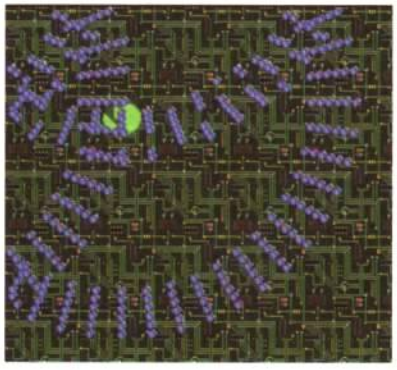
Die DrawSprocket unterstützt auch Double- und Triplebuffering bei der Grafikausgabe. Da wohl nicht jeder Double- und Triplebuffering kennt, hier eine kurze Einführung:

Stellen wir uns ein Spiel mit zwei Sprites vor, die vor einem Hintergrundbild bewegt werden. Für eine schnelle Grafikausgabe ist es von Vorteil, wenn man zuerst das Hintergrundbild als ganzes zeichnet und dann die beiden Sprites darüber ausgibt. Für die nächste Animationsphase gibt es nun zwei Möglichkeiten: entweder wir haben den Hintergrund, den die einzelnen Sprites überdenken, vorher gerettet und restaurieren ihn, bevor wir den Sprite an seine neue Position setzen, oder wir zeichnen das Hintergrundbild einfach komplett neu. Bei ein paar kleineren Sprites lohnt es sich den Hintergrund hinter den Sprites zu retten, aber bei hundert oder mehr Sprites ist der Aufwand zu groß, so daß es einfacher ist, bei jeder Animationsphase den Hintergrund komplett neu auszugeben. Beim C64 oder auch beim Amiga gab es diese Probleme übrigens nicht: die Sprites waren in der Hardware vorhanden, somit brauchte man den Hintergrund nicht vom Programm aus zu retten. Der Nachteil war aber eine Einschränkung in der Größe bei den Sprites. Auch war die Anzahl der Sprites durch die Hardware beschränkt.

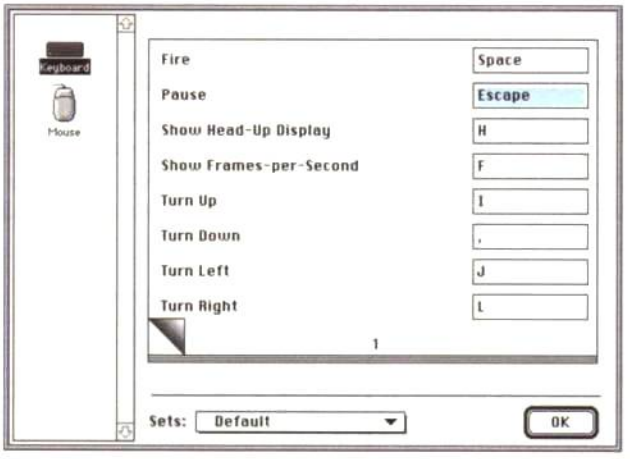
Würde die Ausgabe direkt in den Videospeicher gehen, würden die Sprites deutlich sichtbar flackern, da es immer wieder einen Moment gibt, in dem das Videobild gerade in dem Augenblick dargestellt wird, in dem ein Sprite z.B. erst halb gelöscht ist. Dies führt zum Flackern. Das Flackern kann man auf verschiedene Arten umgehen:

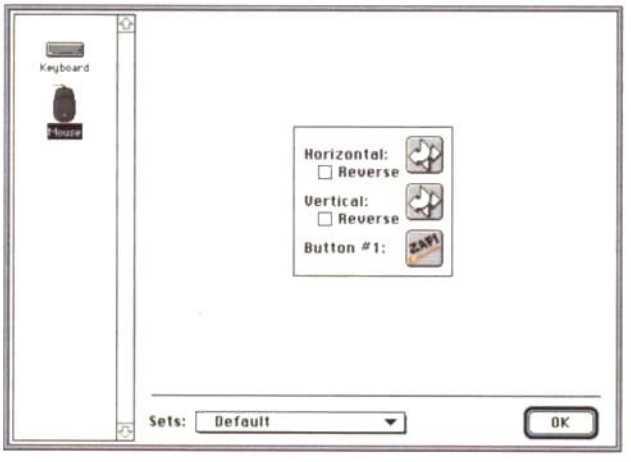
Im einfachsten Fall ist die Grafikausgabe so schnell, daß man sie lediglich mit dem VBL synchronisieren muß. Immer wenn der Elektronenstrahl den Bildschirm rechts unten verlassen hat und zurück zur linken oberen Ecke läuft, hat man etwas Zeit, die bei einfachen Grafiken durchaus für einen Redraw reichen kann. Leider ist diese Zeit sehr, sehr kurz.

Deswegen wird sehr häufig das Doublebuffering genutzt. Hierbei wird eine komplett aufgebaute Grafikseite dargestellt, während man auf einer anderen – verdeckten – Seite die nächste Animationsphase vorbereitet. Ist diese Seite fertig aufgebaut, kann man sie darstellen. Hierbei gibt es zwei Arten der Umschaltung: Page-Flipping, welches von der Grafik-Hardware unterstützt werden muß, erlaubt das Umschalten der Grafik-Hardware auf eine andere Grafikseite. Dieses Verfahren ist sehr schnell und kostet quasi keine Rechenzeit, denn es müssen nur ein paar Register in der Grafik-Hardware umgesetzt werden. Falls diese Umschaltung nicht unterstützt wird, ko-



Viele Sprites auf dem Bildschirm sind für DrawSprocket kein Problem!



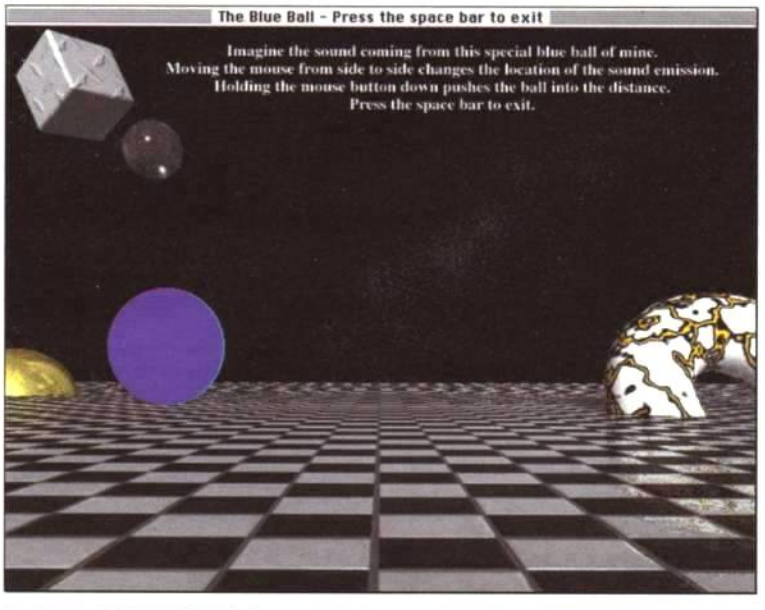


Die InputSprocket stellt eine standardiersierte Form für die Konfiguration von Eingabegeräten zur Verfügung.

piert DrawSprocket mit hochoptimierten Routinen einfach den neuen fertigen – aber noch verdeckten – Bildschirm auf den sichtbaren Bildschirm. Dies sollte jedoch auch mit dem Elektronenstrahl synchronisert werden, sonst kann es – je nach Grafikaufbau – zu sichtbaren Störungen kommen. Da auf die Synchronisation gewartet werden muß, kann man beim Doublebuffering in dieser Zeit nicht die nächste Bildschirmseite aufbauen: die sichtbare Seite ist noch nicht weg und die unsichtbare Seite ist noch nicht kopiert worden. Man muß also warten. Hier hilft Triplebuffering. Während der Wartezeit für das Blitten der Seite 2 auf die Seite 1, kann man auf einer weiteren Seite 3 bereits die übernächste Animationsphase aufbauen. Häufig bringt Tripplebuffering keine Vorteile, da das Spiel auch Zeit für das Berechnen von Spielern etc. braucht, welche häufig aufwendig – sprich: lang – genug ist, um diese Wartezeit zu überbrücken.

Ob Page-Flipping, Double-Buffering in Hardware, direktem Kopieren der Videospeicher oder kompatiblen Kopieren via der Funktion *CopyBits()*, man braucht sich darüber keine Gedanken machen: DrawSprocket wird es optimal machen! Dazu gehört übrigens auch das automatische Skalieren von Grafiken, so kann man ein 320x200-Spiel automatisch auf 640x400 Pixel skalieren lassen.

DrawSprocket wird von Möglichkeiten des Gamma-Fadings ergänzt: das beliebte Aus- bzw. Aufblenden von Grafiken. Das Faden ist übrigens nicht linear, was bedeutet, daß auch dunkle Farben nicht vor den hellen Farben verschwinden. Ferner unterstützt Draw-



In diesem kleinen Beispielprogramm kann man einfach durch Bewegen des blauen Balls die Tonquelle völlig frei bewegen!

Sprocket Overlays (Masken, wie z.B. ein Flugzeug-Cockpit) und Underlays (Hintergrundgrafiken). Man kann die maximale Frame-Rate eines Spiels übrigens vorgeben, so daß ein Spiel nicht mit variablen Geschwindigkeiten läuft, je nachdem, ob die Szene kompliziert oder einfach ist! Zu guter Letzt wird das Debugging unterstützt – was auch dringend Not tut, denn ein komplett auf Schwarz abgedunkelter Bildschirm würde einem Programmierer nicht viel helfen, der Debugger wäre auch unsichtbar.

Apple GameSprockets

Eine Sprite-Engine gibt es übrigens auch: QuickTime 2.5! QuickTime 2.5 hat viele Arten von Sprites und bietet – nebenbei – auch noch die Möglichkeit, die Titelmusik aus MIDI-Daten abzuspielen. QuickTime bedeutet eben viel mehr, als nur einfache Filme abzuspielen ...

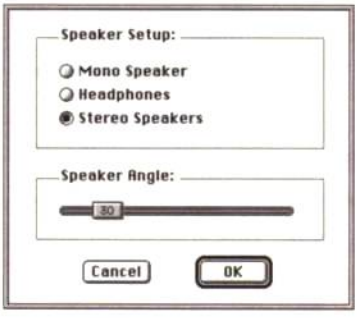
InputSprocket

Die InputSprocket erlaubt die Abfrage von Joysticks, Tastatur und Maus. Vorher mußten die Programme entweder speziell an bestimmte Joysticks angepaßt werden, oder aber die Treiber-Software der Joysticks emulierte einfach Tastendrücke. Das ist nun vorbei! Zudem können über die InputSprocket die Eingabegeräte konfiguriert werden. Dazu müssen die Hersteller der Joysticks allerdings entsprechende Device-Treiber für die Sprocket liefern. Die InputSprocket unterteilt alle Knöpfe etc. an Eingabegeräten in verschiedene Elemente: Buttons, Richtungs-Pad (ein digitaler 8-Wege-Joystick mit Ruhestellung), Achsen mit und ohne Zentrierung (Joystick oder Gaspedal) und Bewegungselemente, die sowohl xy-Werte, als auch Richtungsangaben übermitteln, so daß das Spiel entscheiden kann, was es auswerten will. Das Spiel teilt der InputSprocket beim Start lediglich mit, welche Anforderungen es gerne erfüllt haben möchte, z.B. Bewegungen, Feuertaste, Springen usw. Die InputSprocket kann nun automatisch die Devices entsprechend konfigurieren (der Benutzer kann dies natürlich ändern).

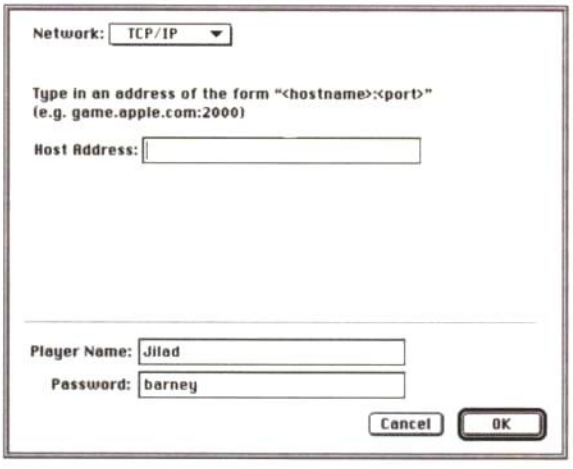
Man kann sich als Programmierer entscheiden, ob man die Eingaben pollen oder als Events mitgeteilt bekommen will. Analoge Eingaben werden üblicherweise gepollt, d.h. dauernd in einer Schleife direkt abgefragt, digitale Eingaben – wie Tastendrücke – läßt man sich als Event schicken. Die Events werden ähnlich *GetNextEvent* in einer Event-Loop abgefragt, jedoch wird nicht mehr zwangsweise *GetNextEvent()* aufgerufen, da diese Funktion Spiele ja kräftig ausbremst.

SoundSprocket

Um eine ansprechene Soundausgabe für sein Spiel zu erzielen, gibt es die SoundSprocket. Sie unterstützt räumlichen 3D-Sound, ähnlich den Dolby-Surround-Effekten im Kino: ein Auto kommt von hinten und fährt links am Spieler vorbei. Ferner kann die SoundSprocket den Doppler-Effekt, Abschwächung von Tonquellen in der Entfernung, Hall in Räumen sowie Positionen von Tonquellen im Raum im allgemeinen simulieren. All dies mit einfachen Stereoboxen! Der 3D-Sound setzt übrigens den Sound Manager 3.2.1 voraus, der jedoch dem GameSprocket-API beiliegt.



Im Ton-Kontrollfeld kann man über diese Erweiterung die Tonausgabe für die SoundSprocket optimieren.



Die Netzwerk-Dialogboxen (oben und unten) sind zwar nicht schön, aber wem sie nicht gefallen, der kann in sein Spiel auch eigene Einstellungen implementieren.

Das High-Level-Interface der SoundSprocket ist übrigens in QuickDraw-3D integriert, setzt es jedoch nicht voraus. Man kann sich das Sound-Modell aber so einfacher vorstellen: jede Tonquelle hat eine Position im Raum und eine Richtung. Der Spieler befindet sich irgendwo in diesem Raum – bewegt sich vielleicht auch noch. Der Gesamteindruck der Tonquellen des Raumes wird komplett von der SoundSprocket errechnet!

NetSprocket

Die NetSprocket verdeckt die schwierige Handhabung der Kommunikation verschiedener Rechner in einem Spiel. Dabei berücksichtigt sie, daß Spieler in einem Spiel durchaus unterschiedliche Funktionen haben können. So kann es Teilnehmer, Zuschauer oder Schiedsrichter geben. Häufig sind Teilnehmer in Teams zusammengefaßt. Ein Schiedsrichter kann nun z.B. eine Message an ein oder mehrere Teams schicken.

So können mit der NetSprocket mehrere Spieler via AppleTalk, TCP/IP (Internet!) und der seriellen Schnittstelle (Modems) vernetzt werden. NetSprocket sorgt auch für eine fehlerfreie Datenübertragung, und dazu gehört auch die Problematik, was passiert, wenn ein Spieler sich plötzlich nicht mehr meldet – sei es, weil das Spiel bei ihm abgestürzt oder weil das Netzwerk zusammengebrochen ist. NetSprocket nutzt in der Version 1.0 übrigens eine Client-Server-Topology, d.h., es gibt einen Server, der Messages an mehrere Clients schickt. Es gibt natürlich auch Spiele, bei denen jeder Rechner sowohl Client als auch Server ist: in MIDI-Maze z.B. berechnet jeder Rechner für sämtliche Mitspieler das Spiel; über das Netz werden nur die Tastendrücke und die Joystick-Positionen übermittelt! Dies könnte man mit der NetSprocket natürlich dadurch lösen, in dem man die Nachricht (Taste wurde gedrückt) einfach an alle Spieler schickt.

Die NetSprocket bietet zudem eine eigene Oberfläche für den Aufbau der Vernetzung, z.B. das Aufnehmen neuer Spieler in ein Spiel. Wem die Oberfläche (Dialoge) nicht gefällt, der kann natürlich etwas eigenes machen.

SpeechSprocket

Die SpeechSprocket basiert auf PlainTalk und enthält eine sprecherunabhängige (englische) Spracherkennung. Die SpeechSprocket ist jedoch speziell an die Bedürfnisse von Spielen angepaßt. So ist die Erkennung zum einen sehr schnell und zum anderen werden nur sehr wenige Vokabeln aus einem vorgegebenen Lexikon erkannt. Gedacht ist die SpeechSprocket natürlich nicht, um den Spieler dazu zu bringen, ununterbrochen „Fire, Fire, Fire, ...“ zu rufen, sondern, um z.B. bei Adventures einen Gegenstand aufnehmen zu können oder ähnliches.

QuickDraw 3D RAVE

QuickDraw 3D RAVE gehört an sich nicht richtig zu den GameSprockets, da es sich vielmehr um das Low-Level-API von QuickDraw 3D handelt. Dieses API ermöglicht es aber eigenen 3D-Engines, wie sie in Spielen bereits häufig vorhanden sind, auf QuickDraw 3D aufzusetzen. Dies hat einen entscheidenen Vorteil: man kann 3D-Beschleuniger-Boards nutzen! Wer übrigens kein Beschleuniger-Board hat, für den hat QuickDraw 3D RAVE einen völlig transparenten Software-Treiber. Allerdings ist QuickDraw 3D auf 32768 Farben optimiert, so daß ein Betrieb in 256 Farben erhebliche Rechen-Power erfordert!

Wie geht es weiter?

Zu den GameSprockets liefert Apple übrigens einige Demospiele, wobei besonders das alte PD-Spiel „Glypha IV“ herausragt. Der Joust-Clone ist bereits seit vielen Jahren auf dem Mac bekannt und im Source verfügbar. Apple hat dieses Spiel nun an die GameSprockets angepaßt, so daß man ein wirklich reales Spiel als Beispielprogramm bekommt!

Ich werde bis zum nächsten Monat hoffentlich ein Teil von MIDI-Maze auf die GameSprockets portiert haben, so daß ich dann auch ein Beispielprogramm mit weiteren Erklärungen nachliefern kann. Bis dahin empfehle ich besonders die Web-Sites von Apple sowie das „Apple GameSprockets Guide“-Buch, welches dem GameSprocket-SDK beiliegt. Bis dahin!

MFR

Apple GameSprockets: http://devworld.apple.com/dev/games/