

WORD

2008.1 From College of Information Science

4

Project MAX
中毒者たち
by WORD 編集部

特集

状況?何が状況だ。俺が状況をつくるのだ。
by ナポレオン

いえ、奴はとんでもないものを盗んでいきましたぞ。
あなたの心です。
by 錢形警部

シューティングゲーム講座
by WORD 編集部

男は度胸!
なんでもためしてみるのさ
by 阿部高和

青春?若いやつらにはもったいないね。
by バーナード・ショー

あきらめたら
そこで試合終了ですよ
by 安西監督

台風と戯れる
by WORD 編集部

WORDはHD-DVDを 応援します号 目次

2	目次
3	Project MAX
15	アナログ回帰
20	台風と戯れる
29	シューティングゲーム開発講座
41	書籍紹介
43	編集後記



文 編集部 いのひろ
ナレーション 編集部 田口トモ ヲ

MAX !

ジョージアマックスコーヒー。加糖練乳を主原料とする、甘く、甘く、甘いコーヒー。その味は、遙か 30 年以上前から*1 受け継がれてきた伝統の味である。これはそのマックスコーヒーに秋休みのすべてを捧げた、ある男達の物語である。

オープニング

かぜのなかのすーぱるー、すなのなかのぎーんがー (r y

アナウンサー A 「Project Max 中毒者達。今回のプロジェクトの舞台は、茨城県つくば市、筑波研究学園都市の中心に位置する、筑波大学です」

アナウンサー B 「ここにあるコーヒー。目を引くデザインですね。ジョージア、マックスコーヒー。加糖練乳の強烈な甘さゆえ、伝説のコーヒーとも呼ばれたこのマックスコーヒー。そんなマックスコーヒーに 2007 年の秋休みを捧げた、ある大学生達の物語です」

2007 年 9 月

筑波大学情報学群情報科学類誌 WORD の編集部員であるいのひろは、あるコーヒーを好きこのんで飲んでいた。「なんて素晴らしいコーヒーなんだ。加糖練乳が主原料だなんて、素晴らしいすぎる」

ジョージア、マックスコーヒー。それは茨城県、千葉県、栃木県でしか販売されていない、幻のコーヒー*2。特徴的なデザイン、そして甘く個性的な味。甘いものが大好きであるいのひろにとって、それは飲料水と同じ感覚だった。

「そうだ、このマックスコーヒーのすばらしさをみんなに教えてあげよう」いのひろは思い立った。すぐに WORD 編集部に行き、友人に勧めた。「すごい甘さなんだよ、まずは一口飲んでみてよ」WORD 編集部員の mitty はいのひろに勧められるがまま、飲んだ。mitty は一口飲んで、「これは素

*2 2006 年 7 月より、東京でも販売しているらしい…

Project MAX -中毒者たち-

晴らしい！なんていう甘さなんだ。私は今までこんな魅力的な飲み物と出会ったことがない。コーヒーと言うよりも練乳を飲んでいるという感覚が、最高だ！」と言った。次に同じく編集部員のかづきおに薦めた。「なんておいしいんだ。もう水なんていらない。僕はこのマックスコーヒーがあれば生きてゆけるよ」大好評だった。

2007年10月

いのひろはマックスコーヒーを第三エリアB棟の下にある自動販売機で購入していた。1本(500ml ペットボトル)140円。売れている為か、他の製品よりも10円高いマックスコーヒーを毎日買うのは、経済的な負担が大きかった。またマックスコーヒーは人気商品であるため、売り切れていることが多かった。「そうだカワチに行こう！」桜にあるカワチ薬局に行つたいのひろは、118円で売られているマックスコーヒーを見つけた。「これを大量に買い込めば、低成本で安定したマックスコーヒーライフを送ることが出来るぞ…」とりあえず5本買った。

2007年11月

2学期の期末テストが始まった。いのひろは情報科学類生の敵、解析学^{*3}に苦しめられていた。「くそっ、テストが終わったら、テストが終わったらだ。マックスコーヒーを大量に買い込んで、みんなでマックスコーヒー料理大会をするんだ」それがProject MAX構想だった。

2007年秋休み

ついにそのときがきた。いのひろは密かに暖めていたProject MAX構想を公にし、WORD編集部1年生+で実行することを宣言した。Project MAXでは以下のメニューが予定されていた。

- MAX 炊き

水炊きならぬMAX 炊き。

MAXな鍋をやっちゃおうということ。

- MAX ラーメン

インスタントラーメンをマックスコーヒーで煮詰める。

MAXなラーメンをやっちゃおうということ。

- MAX せんざい

白玉ならぬマックスコーヒー玉と小豆のマックスコーヒー煮

マックスな和の甘味。

- MAX コーヒープリン

マックスコーヒーを主原料としたプリン。しかしプリンの素を使う。

どうせやるなら、ということで8リットル作ることになった。

マックスな洋の甘味。

これらの料理を作るためには、大量のマックスコーヒーが必要だった。いのひろは悩んだ。「マッ

*3 単位オタヽ(^o^)ノ

クスコーヒーを低価格で大量に購入するには自動販売機ではダメだ…そうだカワチだ！カワチ薬局で箱買はずれば良いのだ！！！」いのひろは編集部員のひなたちゃん（牛久市 18 歳、男性）に車を出してもらい、カワチ薬局へ直行した。

いのひろ「マックスコーヒーをください」

カワチ「！！！あの、マックスコーヒーですか！！！何本でしょうか？」

いのひろ「2 箱！！48 本ください！！」

カワチ「!!!!！」

いのひろ「急いでいるんだ。はやくしないと、はやくしないとマックス分^{*4}が！マックス分が切れそうなんだよー！！！」

カワチ「ジョージアマックスコーヒー、48 本入りました～～！！」

いのひろはマックスコーヒー 2 箱（24 本 * 2 箱）を購入した。

いのひろ「これで、これでやっと Project MAX を実現できるぞ。ふいふいふ、ふはははははん・・・」

2007 年 11 月 29 日

Project MAX 前日。Project MAX におけるお料理の一つである「MAX コーヒープリン」を作るために、いのひろは mitty と共に学内某所にいた。「マックスコーヒーで作る、MAX コーヒープリン。どうせなら MAX にやろうではないか」MAX で MAX な MAX プリンを作るため、ジョージアマックスコーヒー（500ml）14 本、ハウス食品プリンミックス 23 箱、よく鍋料理で用いるような鍋、そして 8 リットルのバケツが用意された。

（右：そのとき待つマックスコーヒー 2 箱、プリンミックス 23 箱、8 リットルバケツ、鍋。下：プリンミックス 23 箱を WORD 編集室で並べてみた。）



いのひろは鍋にすべてのプリンミックス、用意したマックスコーヒー 14 本のうち 9 本を入れた。

*4 マックス分とは、マックスコーヒーを摂取すると獲得できるポイントの事であり、RanRan 分と同じようなものである。マックスコーヒー中毒者の間ではよく“MP”と呼ばれたりする。またマックス分の摂取を怠ると、諸処の禁断症状が現れると言われている。

Project MAX -中毒者たち-

火にかけ約 70 分で粉がすべて溶けるまで暖めながらかき混ぜる。



第三エリアは練乳とコーヒーの香りが混ざった、甘い匂いで満たされた。沸騰させないようにトロトロと煮ていく。すべてのプリンミクスが溶けたことが確認できたら、次はいよいよバケツに移す作業だった。きれいに洗った新品のバケツ（8 リットル）に残しておいた 3 割のマックスコーヒーを入れた。プリンミクス水溶液は冷えると固まってプリンになる。暖めていないマックスコーヒーと混ぜ合わせることによって、MAX コーヒープリンの凝固を開始させる事が出来るのだ。細心の注意を払い、バケツに移した。



すべて移すことができたらお玉でよく混ぜる。そしてここでカラメルの登場である。カラメルとはブッキンプリンで言う、あの色が濃くて特別甘い箇所である。プリンミクスにはカラメルを作るための粉が添付されているので、それも MAX コーヒーで溶いた。割り箸を伝わせてバケツに注入していく。カラメル水溶液はプリン水溶液よりも重いため、勝手にバケツの底に沈んでくれる。これのおかげで、プリンの開発が成功した場合、正にブッキンプリンをお皿にブッキンしたかの如

く、カラメルがプリンの上部に位置するのである。

すべての行程を完了したプリン水溶液はバケツの縁、ギリギリまで来ていた。

MAX コーヒープリンは極めて危険なので放射能標識(?)を印刷し、バケツに貼ることにした。



2007年11月30日

Project MAX 当日。いのひろは朝から買い出しに出かけていた。水炊きならぬ MAX 炊き、MAX ラーメンの材料である。またカワチでは 8.8 リットル入る UNIX Ware (ASVEL 社製) のタッパーを購入。準備は整った。

午後 11 時、Project MAX スタート。2 時間遅れのスタートだった。

最初に取りかかったのは MAX ぜんざいの開発である。しかしながら開発技法の調査不足により^{*5}、この料理はお蔵入りとなってしまった。

次にメインディッシュの MAX 炊き（鍋）である。プリン開発に使った鍋に、ざく切りにした春菊、長ネギ、白菜を放り込む。その上に鶏肉を乗せた^{*6}。Ready for MAX。参加者は合図と共に鍋一杯に MAX コーヒーを投入したのだった。

(右：美味しいそうな鍋)



20 分程度経った頃だろうか、プリン開発時と同じ、甘く、甘ったるいあの匂いが、再び漂い始めたのだ。

*5 小豆は 1 日前から煮ないといけないらしい

*6 作り方が間違っているらしい…

Project MAX -中毒者たち-

参加者 A 「これが、マックスコーヒーの芳醇な香りか！」

参加者 B 「甘い！なんと甘ったるいんだ」

ぐつぐつと鍋が煮えてきた。フタを開け、状態を確認する。そこには暖かいマックスコーヒーに幸せそうにつかる、鶏肉達がいた。紙皿に適当に取り、食す。



いのひろ「!!!!」

参加者「!!!! ...うまい」

甘ったるく、食べたものでは無いと想定されていた鍋は、中毒者達の空腹を満たしたのだ。

参加者「うまい、うまい。結構いけるね」

中毒者達「ははは！みたか！これがマックスヨーヒーの鍋、MAX 鍋じゃ！！！他のどんな液体にも、この味は真似できない。MAX が世界を救うのだー！！！MAX こそ救世主！！」

美味しく煮えた鶏肉は、瞬く間に無くなった。そしてそこには、大量のスープが残った。

いのひろ「うどんの出番だ」おもむろに冷凍庫からうどんを取り出したいのひろは、10 玉すべてを投入した。



当初の構想では、うどん粉からマックスコーヒー麺を作り、めんつゆの代わりにマックスコーヒーを用いる、MAX うどんなるものが考えられていた。しかしながら、うどんを麺から作るのはか

なり難易度が高いことが判明し、また鍋と統合する案が出たため、MAX うどん案もお蔵入りメニューとなつたのであった。

さすがに 10 玉は多すぎたのか、中毒者達の腹は既に満腹であった。その後にはインスタントラーメンをマックスコーヒーで煮る MAX ラーメンが待ちかまえていたが、胃腸と相談した上で、このメニューもお蔵入りとなつた。



うおお、まどろっこしいぜっ！！！^{*7}

(残った MAX 鍋のスープを胃に流し込む mitty)



その後、誰一人として、
mittyの姿を見た者はいなかつた…

MAX コーヒープリンの登場

最後は前日に仕込んだ MAX コーヒープリンである。MAX コーヒープリンはその大きさから、一晩で固まりきるかと懸念されていたが、ある研究室の冷蔵庫を借り、冷やしてもらうことになつていた。そして満を持して冷え切ったバケツが登場。午前中に購入したタッパーを用いてバケツをひっくり返すことにした。

*7 WORD 編集部員が無茶をするときによく用いるかけ声の一つ。前回の無茶は 2007 年 9 月発行の「消えた A 棟の謎号」、「ようこそ、WorDnald へ」に収録されている。

Project MAX -中毒者たち-



完全に冷え切った状態の MAX コーヒープリン

8 kgのプリンはかなり重たいが、ひっくり返すことに成功した。これからバケツを取り外す作業に移るのだが、いくつかの方法が考えられた。

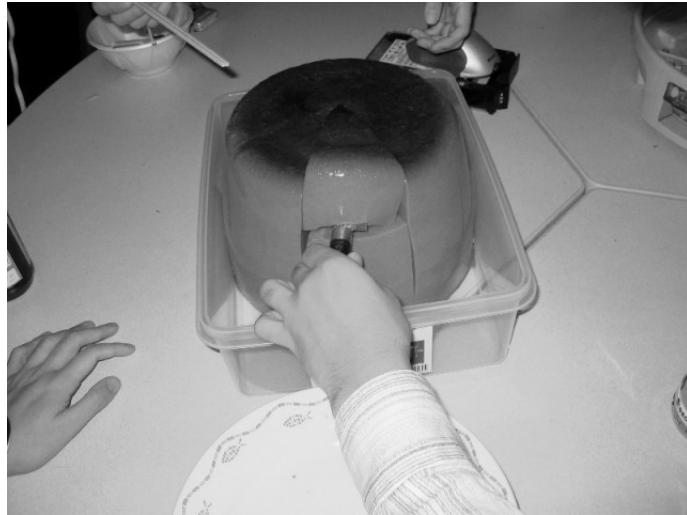
- ・強引にバケツを引っ張りあげる方法
- ・側面から空気を入れてみる方法

中毒者達は、最初は側面から空気を入れる方法でプリンを取り出そうとしたが、プリンが余りにも重たいので、強引にバケツを引っ張ればプリンが落ちるのではないか、ということで前者の方法で取り出すことになった。いのひろは勢いよくバケツを上に引っ張った。バケツからは黒光りする、よく出来たプリンが落ちた。歓声が上がった。



Microsoft Windows NT Workstation のインストールディスクとの大きさの比較

カラメルも非常にきれいにバケツの底に貯まったようである。4年生の先輩曰く「WORD らしくないクライ MAX だな。ナイフを入れたら中が固まって無くて崩れるのではないか」と言った。ナイフを取り出し、切り込みを入れる…崩れない…というか、ナイフの刃渡りが短いのか、プリンがでかすぎるのか、一回で底まで切ることが出来なかった。



ム 力「怯えることはない。こいつは始めから固まっている」

ケーキのように三角に切って、皿に盛った。スプーンで口に運ぶ…甘い。正真正銘の MAX コーヒープリンだった。「これは良い」「これはうまい」「利根コカコーラ・ボトリング^{*8} とハウス食品^{*9} に持ちかければ商品化も夢ではない」大評判だった。



皿に盛った MAX コーヒープリン。上部が黒いのはカラメルが乗っているからである

*8 ジョージアマックスコーヒーの製造元

*9 プリンミクス（プリンの素）の製造元

Project MAX -中毒者たち-

エンディング

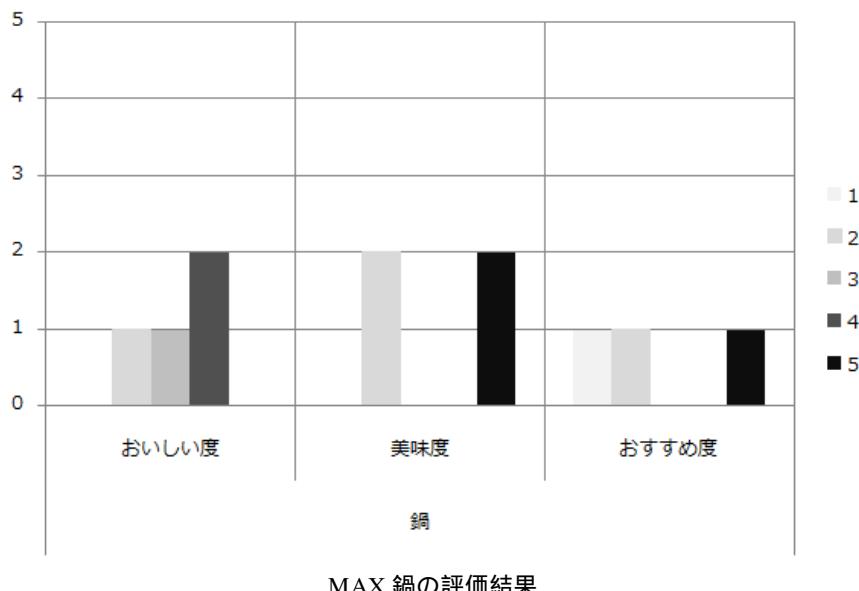
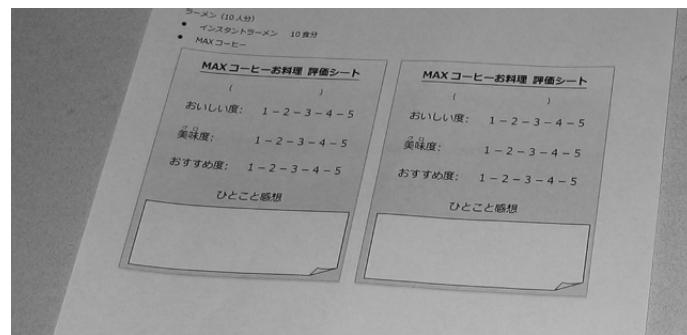
アナウンサー A 「ついに Project MAX のすべての行程を終了した中毒者達は、今後の課題を模索し始めます」

かたーりつぐー、ひともなくー (ry

MAX コーヒープリンが思いのほか好評だったため、Project MAX を年 1 回の恒例行事にしようといひのひろは密かに考えていた。来年は、今年作ることが出来なかつた料理を作ろう。もちろんプリンも作る。それを宣言したとき、反対する人は一人もいなかつた。

まとめ

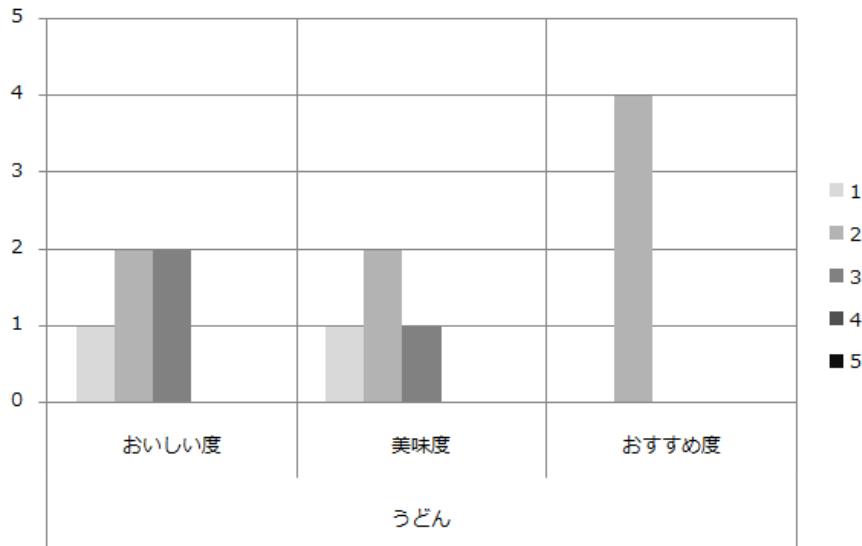
実は写真のようなお料理評価シートを配布して、記入してもらつたのでその結果を掲載しようと思います。この紙には「美味しい度」「美味度」「おすすめ度」を五段階で評価してもらひ（5 の方が良い評価）ひとこと感想を書いてもらひました。



「見える」と「見えない」が別れるような結果に。

ひとこと感想の結果

お肉は大変おいしい
見た目ほどひどくない。というか、つみれは普通においしい
思ったほど悪くない。肉はうまい
空腹時に食べる事をお勧めする
ポン酢につけるとウマー
ポン酢に MAX コーヒーの甘さが適当にあう事がある

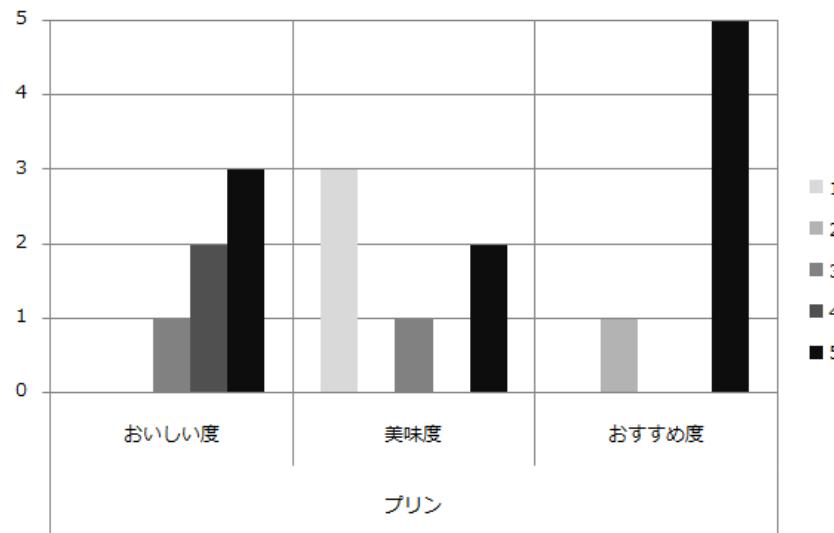


MAX 鍋にうどんを入れたものの評価結果

ひとこと感想の結果

だんだん何を食べているのかわからなくなってきた
見た目は味噌煮込みうどん。味は最悪
ココロが折れそう...
甘いよ。甘い

Project MAX -中毒者たち-



MAX コーヒープリンの評価結果

ひとこと感想の結果

MAX

おいしさ MAX !

食感が良い

うまい。ただ量が…

超濃厚

これは売り込める！

これらの結果から、MAX コーヒープリンがいかに美味であったかが分かると思います。マックスコーヒーと MAX コーヒープリンのおいしさを常人に知ってもらうには学祭で売るしかないですね。これは。

アナログ回帰

文責 かづきお

デジタルに疲れていませんか？

そんなときは、すべての源、アナログへ回帰してみましょう。

アナログの代表格といえば、そう、レコードです。

12インチの円盤に溝を掘り、針で辿ることで音を出す。主格の座をCDに取って代わられたとはい、今でも根強い人気を持つアナログレコード。実際に音を聞いたことのあるという方は少ないかもしれません、実はレコードはCDにも劣らないものすごいパワーを秘めているのです。その良さを少しでも伝えられたらいなと思います。

そんなわけで、筆者はアナログの良さを取材すべく「東海の四天王」が一人、hama氏の自宅へ突撃しました。

hama氏は筆者の高校時代の友人の父上であり、一家そろってオーディオ好きというファミリーの大黒柱です。筆者に自作スピーカーの良さを教えてくれた人物であり、魂の師でもあります。

hama氏のシステムは、プリアンプにパイオニアのC-90、パワーアンプに同M-90。スピーカーはFostex社の限定ユニット^{*1}「FE168ES」を使用した手作りスピーカー「ネッシー Jr.ES」と同社製限定ユニット「6N-FE88ES」を使用したスピーカー「D-88 スーパーフラミング」の二つ。(実はこの他にFE108ESを利用した「スマール・ネッシー」というスピーカーも有るのですが、マトリクス再生^{*2}のリア用^{*3}として設置してあったので音を聞くことはできませんでした)

プリアンプ、パワーアンプというのは、CDプレーヤーから出力された信号を増幅するための機器のことです。プリアンプは、入力切り替えや昇圧などを行います。ボリュームが付いており、パワーアンプに出力する信号を絞ることができます。パワーアンプは、純粹に音の増幅のみを行い、それ以外の機能は付いていません。一般的に、プリメインアンプと呼ばれるプリアンプとパワーアンプが一体化した物が多いのですが、これはパワーアンプから発生する磁界の影響を抑えるためにセパレート化された高級コンポーネントです。

主に使用しているのはネッシー Jr.ESの方。これにはさらにアダプターリング^{*4}「P168」とスパートウイーター^{*5}「T900A」が追加されています。

ちなみにネッシー Jr.ESのジュニアは元々20センチスピーカー用に設計されていた物を縮めて16センチユニットを乗せているから。しかし、縮めたといってもその高さは3メートル弱。天井までそびえ立つウルトラスピーカーです。ちなみにネッシーシリーズのスピーカーは共鳴管と言

*1 スピーカーの実際に振動している丸い部分のこと。自作スピーカーと言ってもこの部分はメーカー製を使います。

*2 ステレオのソースを擬似的に4Chにして再生する方法。

*3 聽く位置より後ろ側に設置されるスピーカーのこと。

*4 ユニットと本体の箱(エンクロージャー)の間にかませる真鍮製のリング。スピーカーの音に真鍮の響きが乗り音がシャープになります。

*5 高音域を専門に受け持つスピーカーのこと。一般にスピーカーは口径が大きくなるほど高い音が再生できなくなるのでスパートウイーターをつけて超高音域まで再生できるようにします。

アナログ回帰

ってあまり市販されていないタイプのスピーカーです（市販スピーカーの大部分はバスレフと呼ばれるタイプ）。名前がネッシーなのも上に大きく突き出した共鳴パイプが特徴的だから。

私はこのネッシータイプよりもスーパーフラミンゴのような「スワン型^{*6}」と呼ばれる三次元立体音道を持ったスピーカーの方が好きなのですが……。

っと、マニアックな説明になってしまい申し訳ない。とにかく、凄いシステムを使っているということです。

肝心のレコードプレーヤーは、MICRO 社製の BL-71 というものです。80 年製ですがきちんと整備されているので今でも現役でいけるそうです。カートリッジは DENON の DL-103SL を使用。そして、比較として CD プレーヤーに TEAC 社製 VRDS-25X を使用。これは故長岡鉄男氏^{*7} が絶賛した VRDS-25XS の一つ前の機種であり、このテのマニアならば必ずと言っていいほど持っている名機です。



外側にあるスピーカーがネッシー Jr.ES

ラックは上からレコードプレーヤー、CD プレーヤー、プリアンプ、パワーアンプ、ウーファー用のモノラルアンプ

^{*6} 長岡鉄男氏が設計したスピーカーの名作。多くのマニアがこのスピーカーに魅了されてきました。

^{*7} 数々のスピーカーを設計したオーディオ評論家。残念ながら 2000 年にお亡くなりになりました。

さて。まずは CD を聞いてみる。

これがまた素晴らしい音を聴かせてくれます。きれいに澄み切った高音。生きしいヴォーカル。ハイスピードで押し寄せる低音。とにかく言葉で表せられないほど素晴らしいんですよ。

ソースは様々な物をかけてみました。J-POP、洋楽、ゲーム音楽、映画のサントラ、ゲテモノ etc。

具体的には、サラ・ブライトマンの「Diva」、Leaf のアレンジアルバム「Pure」、鬼太鼓座の「富嶽百景」、映画「AKIRA」のサントラ、久石譲の「Works」、YMO の「SolidStateSurvivor」等です。ジャンルの統一感がまるで無いのですが気にしないでガンガン再生しました。

ちなみに、筆者のお気に入りは INNOCENCE オリジナルサウンドトラック。

この CD のなかでも、民謡歌手 75 人を集めて収録したという壮大なスケールの合唱曲「傀儡謡（くぐつうた）」が特に私のお気に入りです。その曲をこのシステムで聴くと、きれいにヴォーカルが目の前に広がり、まるで広大なホールにいるよう。そして、それを完全に再生しきるスピーカーの解像度が素晴らしいですね。これをスーパーフラミングで聴くと、「まるで地の底から響くような」(hama 氏) 低音が広がり、それが私のお気に入りでもあるのですが。

と、ここで今回のメインディッシュ、宇多田ヒカルの「FirstLove」の登場です。このアルバム、アナログレコードでも同じ物が売られているのであります。

ちなみに、この CD を選んだのは CD とレコードが発売されているという理由だけではないんです。実はこのアルバムはかなりの高音質を誇る知る人ぞ知る優秀盤なのです。最近のこのテの CD は下手なレコーディングやミキシング、マスタリング等とこのようなシステムで聴くと聴くに耐えない物が多いのですが、このアルバムは歌も素晴らしいのに、その上さらにそれ以外の部分も素晴らしいのです。1000 万枚ヒットはダテじゃない？

アナログ回帰



CD の写真が亡靈のようになってしまった……

さあ、アナログレコードをかけてみよう。

まず、その綺麗さに驚愕しました。レコードというのは、レトロなラジオのような音が出るイメージを持っていたのですが(そんなイメージを持っていたのは筆者だけかもしれません)全然CDとひけを取らない音が飛び出してきたんですよ。たまに「ブツッ」というノイズが入ってしまうことがあります、気になるほどではなく、むしろ味があって良いとまで思われるほどです。

CDでブツブツノイズが入っていたら直ぐに不快になるのにレコードだとそれが許されるのはレコードの持つ不思議な雰囲気のせいなのでしょうか。

肝心の音の方は、暖かみがあってCDより人間が歌っているような感じがします。CDの方がシャープで情報量が多い感じをうけるのですが、これを聞いた後ではむしろ音が詰め込まれているような印象を受けてしまいました。

「これを聴いていると、CDは無くても良いんじゃないかなと思ってしまうよね」とはhama氏の言。

確かにレコードはCDより優れている点が多くあると思います。なにより、12インチのディスクが回転しているというビジョンだけで癒されます。この独特的な雰囲気が音の良さや存在感を引き出しているのではないでしょうか。

アナログレコードを聴いたことのない人は、是非一度体験してみると良いと思います。

ちなみに、どうでもいい話ですが、アンプのボリュームを絞った状態でレコードを再生するとレコード針の所から曲が流れているのが聞こえると思います。とても小さいですが何の曲が流れているのかはっきり分かります。不思議な現象です。

それでは、最後に私の Best 盤を紹介します。



AIR アナログコレクターズエディション

大阪のイベントで限定販売された超貴重なレコード。A 面に鳥の詩、Farewell Song、青空が収録され、B 面に鳥の詩と Farewell Song のリミックスバージョンが入っている。

さらに、樋上いたるさんの書き下ろしピクチャーレーベル仕様。ジャケットも素晴らしいがレコード盤自体に印刷された観鈴ちん。マニアの魂が疼きます。

音質？ そんなのはどうだって良いんです。

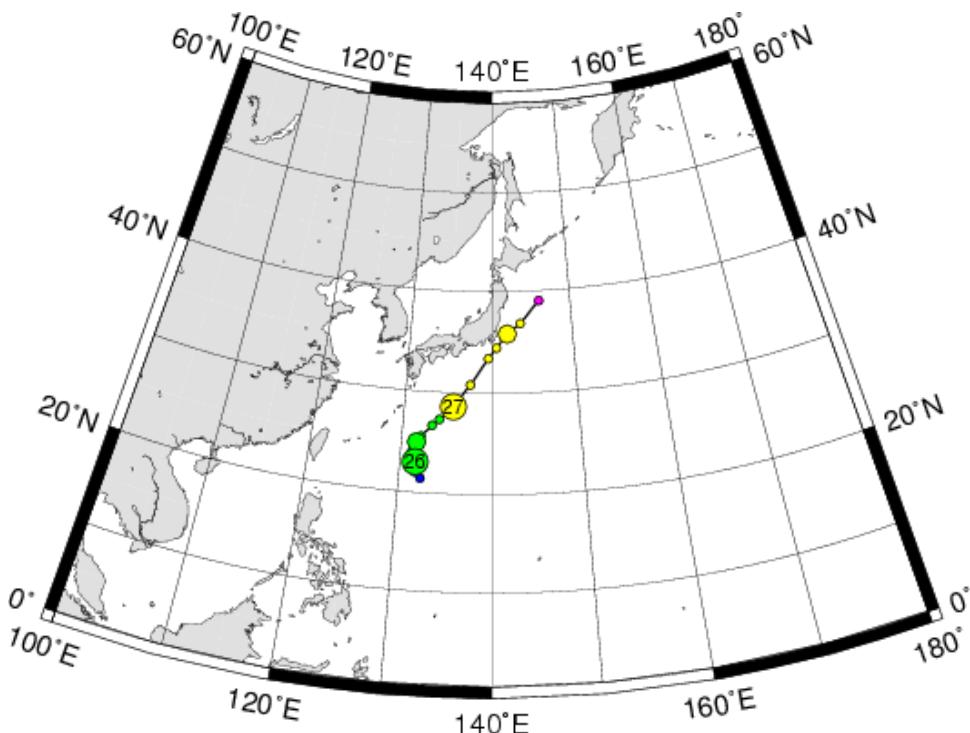
12 インチ盤に描かれたいたるさんの絵が回っていること。それ自体がスバラシイ。

台風と戯れる

文 編集部 あすろんず

通常の三倍の速さ

時は 2007 年 10 月 27 日朝。読者の皆さんには台風 20 号を覚えているだろうか。この台風はかつてないほどの速さで日本列島を駆け抜けた台風だ。その速さはなんと最高時速 95 キロ^{*1}、高速道路を走っているの車とほぼ大差がない。わかりにくいかもしれないで、とりあえずこいつを見てくれ、こいつをどう思う？



発生日時	2007-10-26 00:00:00 UTC	消滅日時	2007-10-27 18:00:00 UTC
継続期間	42 (時間) / 1.8 (日)	最低気圧	980 (hPa)
最大風速	55 (knots)	移動距離	2309 (km)
平均速度	55.0 (km/h) 1319 (km/d)	移動幅	緯度 16.0 度 : 経度 14.8 度

すごく……、はやいです……

*1 北本 朝展 研究室 @ 国立情報学研究所 (<http://agora.ex.nii.ac.jp/~kitamoto/>) にある、デジタル台風 (<http://agora.ex.nii.ac.jp/digital-typhoon/>) のデータベースに基づく情報。以下、本記事で取り扱う情報はすべてこのデータベースに基づく。台風 20 号についての記述は以下のページ。画像の引用も同ページから行った。<http://agora.ex.nii.ac.jp/digital-typhoon/summary/wnp/s/200720.html.ja>

さあ戯れに行くぞ

台風が来るときにはそれなりの準備を行う必要がある。鉢植えが飛ばされないように家の中に入れたり、雨戸を閉めたりするのが普通だろう。ところが、WORD 編集部の AC な編集部員たちは、台風が来るとなぜか元気になり、台風と戯れるのが常となっている。² しかも今回の台風はとびきり速い。ぐずぐずしていてはしゃぎ損ねてしまう。そこで台風の予想進路を調べ、急遽東京湾アクアラインで台風と戯れることが決定し、実行に移された。日時はまだ 10 月 27 日 14 時、きっと間に合うはずだ。（くれぐれも自己責任で行いましょう）

アクアライン

東京湾アクアラインは、東京湾を横断する形で川崎から木更津へ至る高速道路だ。川崎側 9.6km がアクアトンネルで、木更津側 4.4km がアクアブリッジと呼ばれ、その中間地点に海ほたるがある。台風を地肌で感じ取るために、台風の迎撃地点はこの海ほたるということになった。

しかし、この東京湾アクアラインには決定的な弱点がある。風が強くなると通行止め³ になってしまうのだ。14 時の時点で台風は太平洋沖でまだ上陸していない。普通の台風なら余裕で間に合うが、今回は一刻を争う。そんな話をしながら、近くのカレー屋で綿密な計画を練った。

遊ぶ準備

台風と戯れるためには様々な準備が必要だ。高出力のレーザポインタや、カメラ等の映像機器、ビニール傘等の実験器具などである。また今回は、定期的な気象情報の収集のため通信機器を備えたノート PC も持参した。それらの装備を整え、大学を 16 時半の時点で出ることができた。この時点ではまだ台風の暴風域は上陸しておらず、また東京湾アクアラインも通行可能であった。以降、台風で東京湾アクアラインが通行止めになるか、先に到着するか、車と台風の一対一のかけっこが始まった。

れーす

台風とのレースが始まった。今回は常磐道を使わずに、一般道で千葉北インターまで向かうルートを取った。これは常磐道を用いた場合、湾岸線へ乗り入れるため遠回りになってしまふからである。今回は二台の車で向かったが、途中はぐれたり渋滞に巻き込まれながらもなんとか千葉北インターまでたどり着けた。到着直前まで、東京湾アクアラインは通行可能な状態であった。ところが、ブラウザの更新ボタンを押したとき、事態は急変した。

予定変更

17 時 45 分ごろ東京湾アクアラインが閉鎖されたのである。千葉北インターから高速道路に乗る直前で急遽予定を変更。こんなときのために、台風の時の殺伐とした京葉工業地域⁴ の見学という予定も立てていた。このような台風が来ているときに、非常に殺伐とした火力発電所や製油所、製鉄所を見るというまたとないチャンスなのである。

京葉工業地域

*2 過去に台風と戯れた記事については、WORD ベストセレクション III を参照のこと。

*3 風速 20 m を超えると通行止めになる。

*4 化学工業の比率が高く、怪しい薬品を取り扱う施設などが多数ある。

たいふ～ん

さて、台風を前方から迎撃できなくなったので、ゆっくりと下道を走りながらアクアラインへ向かう編集部員一行は京葉工業地域に突入した。そこには、火力発電所や製油所の煙突から炎が絶え間なく噴出しているなどの、殺伐としたすばらしい風景があった。



化学工業プラントの夜景、路面を埋め尽くす雨水が印象的

市原市海釣り公園

その後京葉工業地域を南下し、市原市海釣り公園に到着した。台風も急速に接近していた。そのため、海釣り公園周辺は、30m/s を超えていると思われる暴風が吹き荒れていた。もちろん施設は閉鎖されていた。



ところで台風というと、傘が壊れることがしそうある。しかし、実際に壊れる瞬間を体験したことがない。そこで実際にやってみることにした。手順は簡単、まず傘を広げる。



しかし、どうやら開く方法が駄目だったらしい。^{*5} 傘は一瞬でひっくり返ってしまい、爆殺されてしまった。そこで、もうひとつの傘を風向きに対して 20 度程度傾けて開いてみた。^{*6} その結果、3 秒ほど開いていることができたが、今度は風が強すぎるためか、傘がへし折れてしまった。



*5 風向きに対して 0 度、つまり風向きに向けて開いた。

*6 開いたとき、ちょうど風に対して傘の面が垂直になる形。

たいふ～ん

体感実験

傘を見事木つ端微塵にした後は、やはり地肌で感じるしかない。そう思った編集部員たちは、暴風荒れ狂う中、風除けとなっていた壁から飛び出して行く。実際にこの程度の規模の暴風だと、まるでエアガンで打たれているように顔に当たる風邪が痛い。また、飛んでくる雨がマシンガンのように打ちみだれ、残念ながら5分と持たなかった。

そこで、車内にいながら、もっと台風と戯れることができないかと考えた結果、いろいろと危ないところに突っ込んでいくことになった。



身の危険を感じられる

そんな身の危険を感じる場所のひとつ。夜間での撮影のため非常にわかりにくいが、真ん中の白い線が道路の端である。つまり本来ならこの白い線より下側は道路である。始めのうちは水溜りに比べ路面の割合が大きかった。だんだん進むにつれて^{*7} 水溜りが増え、仕舞いには路面がなくなってしまった。ちなみにこの道は埠頭等にある柵のない道。^{*8} どこまでが水溜りで、どこからが海かは、見た目だけでは判別できなくなってしまった。水溜りも波打っている。これ以上進むと車が浸水して進めなく帰れなくなるので、Uターンして戻ることにした。

*7 道を進むとどんどん海に近づいていく。

*8 写真では写っているが、この後柵はなくなり、海との区切りがなくなる。下手をすると海にまっさかさまである。

突入 アクアライン

そうこうしているうちに、編集部員一行は東京湾アクアライン木更津側の入り口に到着した。途中、休息をとったりしながら進んでいたため、到着間際に通行止めが解除されたのである。ついに東京湾アクアラインに突入するときが来た。

通行止めが解除されたといえど、いまだ強風が吹き荒れるアクアブリッジ。しっかりとハンドルを握らないと風に流されてしまう。そしてついに、編集部員一行は海ほたるに到着することができた。てっきり日が変わっていると思ったら、22時半ごろであった。



違いが良くわからない海ほたる限定バージョン

通行止めになっていたので、店舗もしまっているかと思いきや、普通に店員がいろいろと作業をしていた。また、走り屋や変なカップルなども居り、非常に殺伐としていた。

夜景

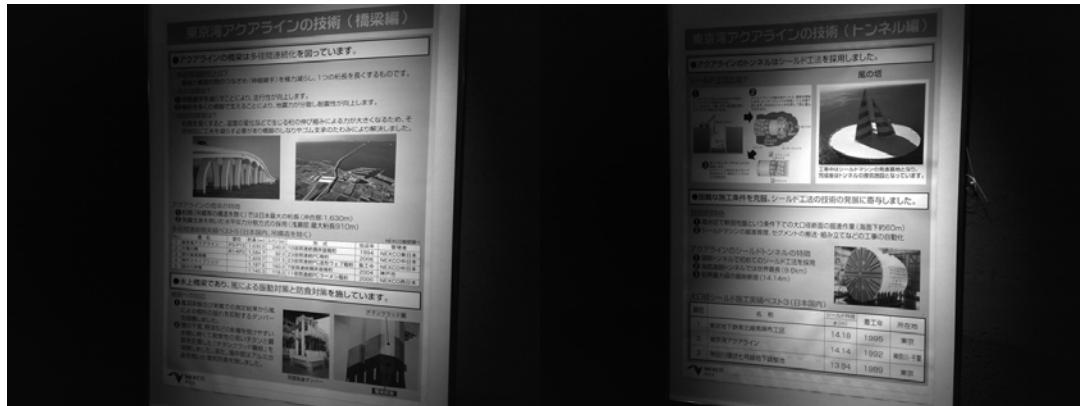
今回の台風は、冒頭でも述べたようにものすごい速さで移動していった。そのおかげか、台風が通り過ぎた後、雲ひとつなくまた空気も澄んでいたため、とても美しい夜景を見ることができた。たぶん普段では見ることができないだろう。



もはや東京湾岸の夜景とは思えない

ところで、海ほたるの中に、アクアラインと海ほたるの成り立ちや工法などの解説をしているブースがある。皆さんも小学校や中学校の社会見学で、このような展示施設に立ち寄ったことがあるかと思う。ここは常時展示されているようだ、深夜にもかかわらず立ち寄ることができた。

*9



アクアラインの技術、トンネル編と橋梁編

意外と知らないことや、映像資料もあり楽しむことができた。また、外にはアクアトンネルを掘削したシールドマシンの刃の現物が展示されていた。

東京湾アクアラインは、その施工時に八機のシールドマシンが使われた。その八機のシールドマシンは、浮島・風の塔^{*10}・海ほたるの三箇所から発進し、24ヶ月かけて東京湾アクアラインアクアトンネルを掘削した。

*9 この常設展示以外に、うみめがね「技術資料館」がある。当日はすでに閉まっていた。営業時間は9:00～18:00。

*10

シールドマシンの直径は14.14メートルと世界最大規模で重量は3200t。なんと6気圧も水圧に耐えられる。

展示されていたのは、そんなシールドマシンの刃の部分になる。シールドマシン解体の影響で、刃の一部が切り取られてしまっていたが、その迫力はなかなかなものである。



夜間はライトアップされているため非常に幻想的

M A X

さらに面白いものはないかと海ほたるの探索を行った結果、恒例のM A Xを発見することができた。MAXコーヒーも最近は都内で買えるようになったので、いつでもMAX分^{*11}を補充することができるはうれしい。

*11 プロジェクトMAXを参照のこと

たいふ～～ん



MAX コーヒー@海ほたる

戯れの後に

さて、今回は台風と様々な形で戯れた。異常な状態のときは、普段見ることができない景色を見れたり、貴重な体験ができる。

ただし、一歩間違えば危険が付きまとうため、実行に移すときはしっかりと予定を立て、万が一のときにも備えておきましょう。そうすれば、どんな異常なときでも楽しむことができると思います。

シュー ティングゲー ム開発講座

文 編集部 yasu haru

目次

1. はじめに	7. プレイヤーの弾の表示
1. 開発環境、開発言語	1. 弾の作成
2. 目標	2. 弾の移動
3. 検証環境	8. 敵とプレイヤーの弾との衝突判定
2. 環境の設定	1. 衝突判定
3. Window の表示	9. 敵の弾の表示
4. プレイヤーの機体の表示	1. 弾の表示
1. Player 構造体の定義	2. 敵の行動
2. プレイヤーの機体の表示	3. 敵の弾とプレイヤーとの衝突判定
5. プレイヤーの機体の移動	10. 最終章
1. タイマーの設定	1. 敵が上から流れてくるようにする
2. キーの取得	2. プレイヤー、敵、弾を画像を使って表示する
6. 敵の機体の表示	3. 画面のちらつきの制御をする
1. 敵のチェインを作る	4. 背景色を変更する
2. 敵の作成	5. プレイヤーの弾に制限をかける
3. 敵の機体の表示	

さて、今回は最終回ということで、実際のシュー ティングゲー ムに近づけていく方法について説明します。前回までの内容で既にシュー ティングゲー ムの骨格となる部分は完成しているので、あとは肉づけを行うだけです。

前回までの記事についてはネット上に掲載してありますので、そちらを参照してください。

シュー ティングゲー ム開発講座 第 1 回

<http://yasuharu.net/word/shooting/1/>

シュー ティングゲー ム開発講座 第 2 回

<http://yasuharu.net/word/shooting/2/>

シュー ティングゲー ム開発講座 第 3 回

<http://yasuharu.net/word/shooting/3/>

前回までは、先に以前までのソースコードを掲載していましたが、今回は、まとまったソースコードの掲載を行わずに、ネット上に最終的なソースコードを掲載するようにします。また、記事中のソースコードについては、ネット上に掲載されているソースコード上の行数を表示してあるので、それを見ながら記事を見てもらえるとわかりやすいです。ソースコードは次の場所に置いてあります。

シュー ティングゲー ム開発講座 第 4 回 ソースコード

<http://yasuharu.net/word/shooting/4/source/>

10. 最終章

1. 敵が上から流れてくるようにする

多くのシュー ティングゲー ムでは、敵がずっと止まっているということではなく、一定の方向やランダムな方向から流れてくることが多いです。そこで、このシュー ティングゲー ムにも上から敵が流れてくるという動作を加えてみます。

以前に RunEnemy 関数という敵の動作を行うための関数を作りました。この関数の中に敵が弾を撃つ動作が入っています。今回はこの関数の中に敵が移動する動作を加えます。同時に、プレイヤーの SetXToPlayer 関数 (SetYToPlayer 関数) に相当する関数が無いのでこの関数も実装します。先にコードを示します。

コード 1 : SetXToEnemy 関数、SetYToEnemy 関数の追加 (242 行目 ~ 263 行目)

```
void SetXToEnemy(Enemy *mEnemy,int value)
{
    if(0 < value && value < mEnemy->mWindowInfo.Width - mEnemy->Width)
    {
        mEnemy->Position.x = value;
    }else
    {
        mEnemy->Position.y = value;
        mEnemy->Destroy = 1;
    }
}
void SetYToEnemy(Enemy *mEnemy,int value)
{
    if(0 < value && value < mEnemy->mWindowInfo.Height - mEnemy->Height)
    {
        mEnemy->Position.y = value;
    }else
    {
        mEnemy->Position.y = value;
        mEnemy->Destroy = 1;
    }
}
```

コード 2 : コード 1 で追加した関数の呼び出して敵を移動させる (340 行目)

```
SetYToEnemy(mEnemy,mEnemy->Position.y + mEnemy->Speed);
```

コード 1 で、SetXToEnemy 関数、SetYToEnemy 関数を実装します。コードはプレイヤーの弾の実装と変わりません。コード 2 では、コード 1 で作った関数を呼び出して、敵を下に移動させる処理が書いてあります。現在の座標に対して、Speed 分下方向に進めていきます。

それでは、これを実行してみましょう。最終回の今回は、元となる部分は完成しているので、少し変更を加えただけでも動作は大きく変わります。どうやったら、どう変わるのが、いろいろといじってみると面白いので、どんどん試してみましょう。

さて、実行してみると、どこからともなく敵が出てきて、下の方に移動してくると思います。敵が弾を打っても、しっかりとプレイヤーを狙っていて問題なさそうです。ただ、今ままだとどこから出てくるかわかりません。これは好みの問題なのですが、画面の一番上から敵が移動してきた方が自然だと思ったので、そのように変更してみます。

弾を生成する箇所のコードを変更します。今まででは、コード 3 のように X・Y 座標を乱数で、決定していました。画面の一番上から敵が出現するようにするには、Y 座標を画面の上方に固定して、X 座標だけ乱数で求めるようにします。コード 4 では、Y 座標の値を 5 として、画面の上方から敵が出現するようにしています。

コード 3：変更前のコード

```
Position.y = rand() % mWindowInfo.Height;
Position.x = rand() % mWindowInfo.Width;
```

コード 4：変更後のコード（488、489 行目）

```
Position.y = 5;
Position.x = rand() % mWindowInfo.Width;
```

以上で、画面の上方から敵が流れてくるようにすることができます。

これらを応用して、敵がランダムな方向に動いたり、プレイヤーを追跡したりすることができます。また、敵の生成時のスピードをランダムにすることで、もっと変化をつけさせることができます。

2. プレイヤー、敵、弾を画像を使って表示する

次に、プレイヤー、敵、弾のそれぞれの画像を使って表示します。画像はビットマップファイルを読み込んで、それを画面上に描画するようにします。

手順としては、まず、画像をロードするためにデバイスコンテキストというものを作成します。デバイスコンテキストとは、ディスプレイやプリンタなどの出力するためのデバイスを仮想的に取り扱うための仕組みです。このデバイスコンテキストの上に描画の処理などを加えることでディスプレイに描画を行うことができます。これは以前にも使っていて、BeginPaint 関数で取得して、TextOut 関数などを使って描画するときに用いた hdc という変数がまさにそれです。このときは、ディスプレイのデバイスコンテキストを使用しました。

ここでは、メモリ上にデバイスコンテキストを作成します。画像をロードするときには、このデバイスコンテキストに対してロードを行います。このようにして、画像をメモリ上に保持しておきます。描画をするときには、画像を読み込んであるデバイスコンテキストからディスプレイのデバイスコンテキストへ画像をコピーします。

手順を説明したところで、次に使用する関数について説明をします。まず、デバイスコンテキストを作成する部分です。デバイスコンテキストを作成するには、CreateCompatibleDC 関数を使用します。

```
HDC CreateCompatibleDC(
    HDC hdc // デバイスコンテキストのハンドル
);
```

(引用元)

http://msdn.microsoft.com/library/ja/default.asp?url=/library/ja/jpgdi/html/_win32_createcompatibledc.asp

この関数では、引数のデバイスコンテキストのハンドルと互換性のあるデバイスコンテキストを作成して、そのハンドルを返します。互換性については後から説明をします。この関数で作成したデバイスコンテキストは、必ず DeleteDC 関数で削除する必要があります。

シュー ゲーム開発講座

```
BOOL DeleteDC(  
    HDC hdc // デバイスコンテキストのハンドル  
);
```

(引用元)

http://msdn.microsoft.com/library/ja/default.asp?url=/library/ja/jpgdi/html/_win32_deletedc.asp

次に、このデバイスコンテキストに対して画像ファイルのロードを行います。ロードを行うには、LoadImage 関数を使用します。

```
HANDLE LoadImage(  
    HINSTANCE hinst, // インスタンスのハンドル  
    LPCTSTR lpszName, // イメージの名前または識別子  
    UINT uType, // イメージのタイプ  
    int cxDesired, // 希望する幅  
    int cyDesired, // 希望する高さ  
    UINT fuLoad // ロードのオプション  
);
```

(引用元)

http://msdn.microsoft.com/library/ja/default.asp?url=/library/ja/jpwinui/html/_win32_loadimage.asp

この関数では、引数に情報を渡してやるとビットマップへのハンドルが返ってきます。帰ってきたハンドルをデバイスコンテキストに関連づけします。こうすることで、メモリ上のデバイスコンテキストから画像を扱えるようになります。デバイスコンテキストにビットマップなどを関連づけするためには、SelectObject 関数を使用して関連づけをします。

```
HGDIOBJ SelectObject(  
    HDC hdc, // デバイスコンテキストのハンドル  
    HGDIOBJ hgdiobj // オブジェクトのハンドル  
);
```

(引用元)

http://msdn.microsoft.com/library/ja/default.asp?url=/library/ja/jpgdi/html/_win32_selectobject.asp

この関数では、引数の hdc に対して引数の hgdiobj を関連づけます。

LoadImage 関数で取得したハンドルについても CreateCompatibleDC 関数で取得したデバイスコンテキストハンドルと同じように、解放する必要があります。このときに使用する関数は、DeleteObject 関数を使用します。

```
BOOL DeleteObject(  
    HGDIOBJ hObject // グラフィックオブジェクトのハンドル  
);
```

(引用元)

http://msdn.microsoft.com/library/ja/default.asp?url=/library/ja/jpgdi/html/_win32_deleteobject.asp

デバイスコンテキストやビットマップのハンドルは、不要になった時点で削除すればいい

ので、今回のシュー タイ ング ゲームではプログラムが終了する段階で削除することにします。さて、いろいろと関数が出てきてちょっとわかりづらかったかと思います。順を追って表してみると次のようになります。

1. CreateCompatibleDC 関数でデバイスコンテキストを作成。
2. LoadImage 関数でビットマップのハンドルを取得。
3. SelectObject 関数でデバイスコンテキストにビットマップを関連づけする。
4. (シュー タイ ング ゲームを行って、プログラムが終了する)
5. DeleteDC 関数でデバイスコンテキストを削除。
6. DeleteObject 関数でビットマップを削除。

それでは、手順の説明が終わったところでデバイスコンテキストにビットマップをロードするまでのコードを見ていきましょう。ここでは、ひとまずプレイヤーの機体の画像だけロードすることにします。

コード 5：変数の定義 (79、80 行目)

```
HDC hPlayer;
HBITMAP hPlayerBitmap;
```

コード 6：デバイスコンテキストの作成 (367 行目 ~)

```
hdc = GetDC(hWnd);
hPlayer = CreateCompatibleDC(hdc);
hPlayerBitmap = (HBITMAP)LoadImage(
    ghInst,"player.bmp",IMAGE_BITMAP,0,0,LR_LOADFROMFILE);
SelectObject(hPlayer,hPlayerBitmap);
ReleaseDC(hWnd, hdc);
```

コード 7：プレイヤー画像を保持しているデバイスコンテキストのハンドルを設定 (401 行目)

```
mPlayer.hImage = hPlayer;
```

コード 8：オブジェクトの解放 (640、641 行目)

```
DeleteObject(hPlayerBitmap);
DeleteDC(hPlayer);
```

コード 5 では、グローバル変数にデバイスコンテキストハンドルとビットマップハンドルを定義しています。

コード 6 で、デバイスコンテキストの作成と画像のロードを行っています。CreateCompatibleDC 関数の引数で指定するデバイスコンテキストハンドルは、ディスプレイのデバイスコンテキストハンドルを使用します。最終的な描画先がディスプレイなので、ディスプレイと互換性を取るようにします。

LoadImage 関数の引数について説明をします。2 番目の引数は、画像のリソース名かファイル名かを指定できます。ここでは、ビットマップファイルを読み込むのでここにファイルのパスを記述します。ファイルのパスは、相対パス、絶対パスのどちらでも指定でき、ここでは相対パスで "player.bmp" というファイルを設定しています。相対パスで指定した場合は、プログラムの実行時のカレントディレクトリがどこになるか、ということを意識する必要があります。Visual Studio 2005 のデバッグ時のカレントディレクトリは、ソリューションファ

シュー ティング ゲーム 開発 講座

イルが入っているディレクトリの debug というフォルダがカレントディレクトリになります。ですので、プログラムが実行される際は、debug というディレクトリの中の"player.bmp"というファイルが参照されます。もし、場所がわからないようでしたら、絶対パスで指定してみてください。絶対パスを指定する際には、パスに"\\"記号が入っているのでエスケープする（\"を\"\"とする）のを忘れないようにしてください。3番目の引数には、イメージのタイプを指定します。今回は、ビットマップファイルを読み込むので「IMAGE_BITMAP」というフラグをセットします。4番目、5番目の引数についてですが、これらの引数では、読み込む画像の幅と高さを指定します。それぞれの引数に0を指定することで、ビットマップファイルの幅と高さが自動的に設定されます。6番目の引数には、ファイルからロードすることを示す「LR_LOADFROMFILE」を指定します。

LoadImage 関数でビットマップファイルのロードする方法を説明したところで、読み込むファイルがないとどうにもならないのでビットマップファイルを作ります。ビットマップファイルの作成はペイントなどを使って行ってください。ビットマップファイルのサイズは 16 × 16 にします。描画を行うときに透過をさせたい部分は白色を指定しておいてください。これについては、後から説明をします。保存先は、LoadImage 関数のパスの指定にあうような場所に保存してください。

ロードしたビットマップは、SelectObject 関数を使ってメモリ上のデバイスコンテキストで選択します。コード 7 で、プレイヤーの情報を保持している構造体のメンバにハンドルを渡しています。これで、メモリ上にビットマップファイルをロードできました。

次に、実際に描画する部分を説明します。描画をするには TransparentBlt 関数を使用します。この関数は、透過を行った上で描画を行うことができます。ちなみに、この関数は Windows 2000/XP/Vista でしかサポートされません。関数の定義は次のようになっています。

```
BOOL TransparentBlt(
    HDC hdcDest,
    int nXOriginDest,
    int nYOriginDest,
    int nWidthDest,
    int nHeightDest,
    HDC hdcSrc,
    int nXOriginSrc,
    int nYOriginSrc,
    int nWidthSrc,
    int nHeightSrc,
    UINT crTransparent
);
```

(引用元)

http://msdn.microsoft.com/library/ja/default.asp?url=/library/ja/jpgdi/html/_win32_transparentblt.asp

関数の定義だけではわかりづらいので、実際のコードも示します。実際のコードは、前回までの でのプレイヤー表示の部分を今回のコードに差し替えます。

コード 9：変更前

```
TextOut(hdc,mPlayer.Position.x,mPlayer.Position.y," ",2);
```

コード 10：変更後（432、433 行目）

```
TransparentBlt(hdc,mPlayer.Position.x,mPlayer.Position.y,mPlayer.Width,mPlayer.Height,
mPlayer.hImage,0,0,mPlayer.Width,mPlayer.Height,0xFFFFFFFF);
```

1 番目の引数の `hdcDest` には、転送先のデバイスコンテキストハンドルを指定します。今回の場合は、直接ディスプレイに描画を行うので、転送先はディスプレイのデバイスコンテキストハンドルとなります。また、6 番目の引数は、転送元のデバイスコンテキストハンドルを指定します。ここでは、画像を読み込んだメモリ上のデバイスコンテキストから転送を行うので、プレイヤーの構造体に格納されているハンドルを渡します。一番最後の引数は、この引数の値で透過色を指定することができます。今回の場合、白色を透過色にするので、16進数のカラーコードで白を表す `0xFFFFFFFF` を指定しています。

`TransparentBlt` 関数を使用するためにライブラリを指定する必要があります。ライブラリとは特定の機能を持ったプログラムなどが再利用可能な状態でまとめられているファイルのことをいいます。ライブラリには、複数の関数が入っており、一部の `TransparentBlt` 関数のような関数ではコンパイル時にライブラリを指定する必要があります。`TransparentBlt` 関数では、「`msimg32.lib`」をライブラリに指定する必要があります。ライブラリに指定する方法は、使っているコンパイラごとに異なります。ここでは Visual Studio 2005 での指定方法について説明をします。

Visual Studio 2005 では、プロジェクトファイルのプロパティに指定する方法もあるのですが、ソースコード中にライブラリを指定する方法もあります。後者の方が簡単なので、そちらの方を説明します。ファイルの先頭に次のコードを追加してください。

コード 11：ライブラリの追加（2 行目）

```
#pragma comment(lib, "msimg32.lib")
```

これで、ライブラリの追加ができます。

コードが書き終わったら、実際に実行してみましょう。画像が表示されてキーを押すと動いてくれれば成功です。もし、機体が黒い四角として表示される場合は、画像ファイルが正常に読み込まれていない可能性があります。ファイルの設置位置を見直してみるか、無理そうなら、絶対パスで記述してみてください（絶対パスを記述するときはエスケープシーケンスに気をつけてください）。このコードでは、プレイヤーの機体だけでしたが、同じようにして敵の機体や弾の画像なども簡単に変えられます。こちらも試してみてください。

3. 画面のちらつきの制御をする

ゲームをプレイしていてプレイヤーを動かしたりしていると、ちらついて見えることがあります。ここではこの直し方について説明をします。

ちらつきを制御するためには、メモリ上に仮想的なディスプレイを作り、そこに描画の処理を行って、描画が完了したら実際のディスプレイに描画を行います。これを、ダブルバッファリングといいます。一端、メモリ上のディスプレイに描画を行ってから、実際のディスプレイに描画を行うことで、画面が切り替わる時間が短くなりちらついて見えることが無くなります。

それでは、具体的にどのようにしてダブルバッファリングするのか説明していきます。まず、メモリ上に仮想的なディスプレイを作るには、先ほど出てきた互換性のあるデバイスコ

シュー ゲーム開発講座

ンテキストを作成する CreateCompatibleDC 関数を使います。これだけでは、このデバイスコンテキストに対してまだ描画ができません。これ加えて、互換性のあるビットマップを作成して、デバイスコンテキストに関連づけます。これは、CreateCompatibleBitmap 関数を使用します。この関数は、次のように定義されています。

```
HBITMAP CreateCompatibleBitmap(
    HDC hdc,           // デバイスコンテキストのハンドル
    int nWidth,        // ピクセル単位のビットマップの幅
    int nHeight        // ピクセル単位のビットマップの高さ
);
```

(引用元)

http://msdn.microsoft.com/library/ja/?url=/library/ja/jpgdi/html/_win32_createcompatiblebitmap.asp

引数には、高さと幅を指定します。

ダブルバッファリングを行う際には、まず、メモリ上のデバイスコンテキストに描画を行います。そして、全部のデータが描画し終わったら実際のディスプレイのデバイスコンテキストにコピーします。コピーする際には、次の関数を使用します。

```
BOOL BitBlt(
    HDC hdcDest,      // コピー先デバイスコンテキストのハンドル
    int nXDest,       // コピー先長方形の左上隅の x 座標
    int nYDest,       // コピー先長方形の左上隅の y 座標
    int nWidth,        // コピー先長方形の幅
    int nHeight,       // コピー先長方形の高さ
    HDC hdcSrc,       // コピー元デバイスコンテキストのハンドル
    int nXSrc,         // コピー元長方形の左上隅の x 座標
    int nYSrc,         // コピー元長方形の左上隅の y 座標
    DWORD dwRop // ラスタオペレーションコード
);
```

(引用元)

http://msdn.microsoft.com/library/ja/default.asp?url=/library/ja/jpgdi/html/_win32_bitblt.asp

この関数では、hdcDest に指定したデバイスコンテキストから hdcSrc に指定したデバイスコンテキストへコピーを行います。最後のラスタオペレーションコードについてですが、SRCCOPY を指定します。この値を指定するとそのままコピーすることになります。他のラスタオペレーションコードについては、引用元のページに書かれているのでそちらを参照してください。

それでは、実際のコードを見てみましょう。

コード 12：変数の定義（348、349 行目）

```
HDC hPlane;
HBITMAP hBitmap;
```

コード 13：デバイスコンテキストの作成（426 行目～428 行目）

```
hPlane = CreateCompatibleDC(hdc);
hBitmap = CreateCompatibleBitmap(hdc,mWindowInfo.Width,mWindowInfo.Height);
SelectObject(hPlane,hBitmap);
```

コード 14：メモリ上のデバイスコンテキストに描画（432、433 行目）

```
TransparentBlt(hPlane,mPlayer.Position.x,mPlayer.Position.y,mPlayer.Width,mPlayer.Height,
mPlayer.hImage,0,0,mPlayer.Width,mPlayer.Height,0xFFFFFFFF);
```

コード 15：ディスプレイのデバイスコンテキストへの描画と解放（467 行目～471 行目）

```
BitBlt(hdc,0,0,mWindowInfo.Width,mWindowInfo.Width,hPlane,0,0,SRCCOPY);
```

```
EndPaint(hWnd,&ps);
DeleteDC(hPlane);
DeleteObject(hBitmap);
```

コード 16：再描画をするための関数（610 行目）

```
InvalidateRect(hWnd,NULL,TRUE);
```

コード 13 で、CreateCompatibleBitmap 関数で作ったビットマップは、プレイヤーの機体の画像をロードしたときと同じように SelectObject 関数を使ってデバイスコンテキストに割り当てるようになります。また、作成したデバイスコンテキストなどは、必要が無くなった時点で解放します。

コード 16 では、InvalidateRect 関数の最後の引数を FALSE に変えました。TRUE の場合は、再描画を行う際に背景を消去しますが、FALSE の場合は、背景を消去しません。

これで、ちらつくことは無くなりました。試しに実行してみましょう。

実行してみると、確かにちらつきが無くなっているのはわかるかと思います。ただ、背景の色が黒（あるいは、他の色）になってしまっています。次のところでは、この背景色の変え方について説明をします。

4. 背景色を変更する

背景色を変更するためには、いくつか方法があります。単純に四角形を描画する関数を使って背景を塗りつぶす方法や、画像をロードしてそれを背景として使用する方法があります。ここでは、後々の応用が利くという点で画像をロードしての背景色の変更について説明をします。といっても、既に画像を読み込んで描画する方法は説明しているので、それを応用するだけです。

まず、背景として設定するための画像を作成しましょう。作成する画像のサイズは、640 × 480 で作成して、プレイヤーの機体の画像を保存した場所と同じ場所に保存します。ファイル名は、"back.bmp"としておいてください。

次に、プレイヤーの機体の画像を読み込んだときと同じようにして、ビットマップファイルを読み込みます。読み込んだら、それをメモリ上のデバイスコンテキストへ関連づけしておきます。ここまででは、プレイヤーの機体を読み込んだときと同じです。

読み込んだ画像はダブルバッファリングを行うときに一番最初に描画しておきます。そうすると、最後にディスプレイの再描画を行ったときに背景として見えます。実際にコードを見てみましょう。

コード 17：変数の定義（79、80 行目）

```
HDC hPlayer,hBack;
HBITMAP hPlayerBitmap,hBackBitmap;
```

シュー ティング ゲーム 開発 講座

コード 18 : デバイスコンテキストの作成とビットマップのロード (385 行目 ~ 387 行目)

```
hBack = CreateCompatibleDC(hdc);
hBackBitmap = (HBITMAP)LoadImage(
    ghInst,"back.bmp",IMAGE_BITMAP,0,0,LR_LOADFROMFILE);
SelectObject(hBack,hBackBitmap);
```

コード 19 : ディスプレイに描画 (430 行目)

```
BitBlt(hPlane,0,0,mWindowInfo.Width,mWindowInfo.Width,hBack,0,0,SRCCOPY);
```

コード 20 : オブジェクトの解放 (652、653 行目)

```
DeleteObject(hBackBitmap);
DeleteDC(hBack);
```

前半部分は、プレイヤーの機体を読み込む部分と変わりありません。コード 19 の部分では、背景画像をダブルバッファリング用のデバイスコンテキストに描画しています。BitBlt 関数も先ほど説明したものです。

それでは、実行してみてください。先ほど作成した画像と同じ背景が表示されているでしょうか？表示されていたら成功です。背景の画像を編集することで好きな背景を表示することができます。

5. プレイヤーの弾に制限をかける

プレイヤーが弾を打つとき、キーを押しっぱなしにしておくとビームのように大量に弾が出て、簡単に敵が倒せてしまいます。これでは、面白くないので 0.1 秒間に一回だけ弾を撃てるように変更しましょう。

これを実装する方法について説明します。まず、最初に弾を打った時間を保持しておきます。その後、次にスペースキーが押されたときに時間を比較して、0.1 秒経っているかどうか確認します。時間が経っていたら弾を発射して、先ほど保持してあった時間を今の時間に書き換えます。これだけの処理で制限がかけられます。

また、一度に撃てる数も変更してしまいましょう。これについては、プレイヤーの弾を保持している配列の要素数を変えるだけで簡単にできてしまいます。

それでは、コードを見てみましょう。

コード 21 : ライブラリの指定 (1 行目)

```
#pragma comment(lib, "winmm.lib")
```

コード 22 : プレイヤーの弾の数の変更 (16 行目)

```
#define MAX_PLAYER_BALL 5
```

コード 23 : 弾を撃った時間を保持する変数の定義 (359 行目)

```
static time_t CreateBallTime;
```

コード 24 : 変数の初期化 (420 行目)

```
CreateBallTime = timeGetTime();
```

コード 25 : 弾を撃つかどうかの判定 (527 行目)

```
if(timeGetTime() - CreateBallTime > 100)
```

コード 26 : 弾を撃ったときの時間の設定 (545 行目)

```
CreateBallTime = timeGetTime();
```

コード 24 では、ゲーム開始時の時間を取得しています。時間を取得するには、timeGetTime 関数を使用しています。乱数の説明の時には time 関数を用いて時刻を取得しましたが、今回は、ミリ秒単位で時間を取り得する必要があるので、この関数を使用しています。この関数の引数などは次のようになっています。

```
DWORD timeGetTime(VOID);
```

引用元

http://msdn.microsoft.com/library/ja/default.asp?url=/library/ja/jpmldtimd/html/_win32_timegettime.asp

この関数では、システムが起動してから経過した時間をミリ秒単位で取得することができます。timeGetTime 関数もライブラリが必要になります。コード 21 のように "winmm.lib" を指定するようにしてください。

コード 25 で、弾を撃つときの時間と前回撃ったときの時間を比較して、経過時間が 100ms より大きかった場合に弾を発射するようにしています。撃った後は、コード 26 の部分で弾を撃った時間を更新することを忘れないようにしましょう。

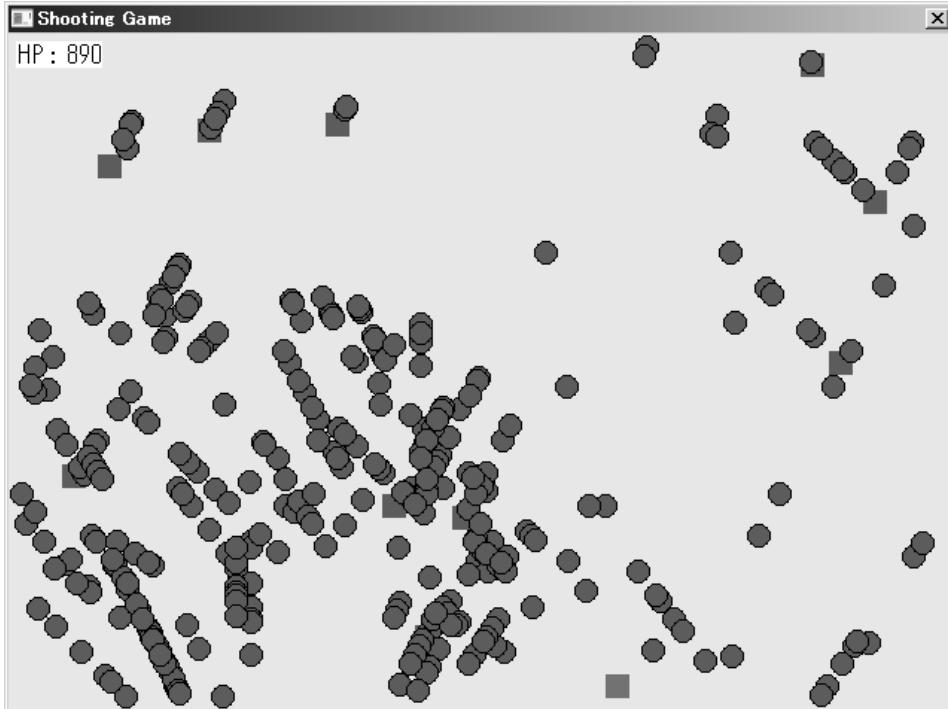
終わったら実行してみましょう。弾が連続で発射できないようになっているはずです。

シューティングゲーム開発講座

最後に

4回にわたる長い記事を見ていただき、ありがとうございます。説明不足でわかりづらいところなどもあったかもしれません。ただ、この記事を読んでゲームのプログラムはこんな風に動いているということがわかつていただけたり、ゲームプログラミングに興味を持っていただければ幸いです。

最後に、完成したシューティングゲームのスクリーンショットを載せておきます。



このシュー팅ゲームは、たった 700 行程度のソースコードでできています。ちょっと勉強をするだけでできるのでどんどん挑戦してみてください。

今回の記事については、こちらでも公開します。また、記事中に出てきたソースコードについては、章ごとにダウンロードできるようになっています。

Web ページ :<http://yasuharu.net/word/shooting/4/>

間違いの指摘などありましたら、以下の連絡先まで遠慮無く連絡してください。

メールアドレス :word@yasuharu.net

前回の記事の訂正について。

前回の記事で間違いがありました。申し訳ありません。

前回の記事の p38 ページのコード 15 の部分を次のように修正してください。

(訂正前) static Ball *mEnemyBall[MAX_PLAYER_BALL];

(訂正後) static Ball *mEnemyBall[MAX_ENEMY_BALL];

訂正を行わないと、今回のコード 22 のコードに変更した際にエラーがでてしまいます。

書籍紹介

文 編集部 ひなたぼこ

ゲームプログラミングやってみないか？

「yasuharu シューティングゲーム講座^{*1}」でゲームプログラミングはバッチリだ！あとは俺様の超絶グラフィックスを美しく華麗に描画させたいな。あ、いまどきはフルヴォイスなんて当たり前だからな、俺の美的ボイスで一人多役のフルヴォイスに対応させねば…しかしどうやってやるんだ？」^{*2} というクリエイターな方は、今回紹介する書籍を参考にしてみてはいかがでしょうか。

DirectGraphics の入門書



この本では、DirectX^{*3}に含まれる機能の一つである DirectGraphics の初期化方法から、エフェクト効果の基礎応用までの説明が丁寧に一通り書かれています。

3D プログラミングのための基本的な知識や数学の説明も含まれているので、3D プログラミングや DirectGraphics の基本を重点的に勉強をしたい方にはこちらの書籍をオススメします。

ただ、この本はあくまで DirectGraphics の入門書なので、DirectX のサウンド機能や入力機能を扱いたい方には、次に紹介する DirectX9.0 実践プログラミングが良いでしょう。

書籍名：DirectX ゲームグラフィックスプログラミング ver.2.1 Vista 対応

著者名：N2 Factory

発 行：ソフトバンククリエイティブ

ISBN-13：978-4797341874

値 段：2940 円

ペー数：360 頁

発行日：2007/6/29

DirectX9.0 の入門書

DirectX9 実践プログラミングでは、DirectX の各機能 DirectGraphics, DirectSound, DirectMusic, DirectInput, DirectPlay, DirectShow についてざっと書かれています。

ゲーム開発に利用する程度であれば、DirectX の基本は全て網羅しているので、これから DirectX を学んでみようとする方にオススメです。

*1 2007 年から WORD で連載されているシューティングゲームの開発講座。

*2 マイクロソフトが提供しているマルチメディア拡張 API 群、ハードウェアを統一的方法で操作 することが出来るため、フルパフォーマンスのマルチメディア処理を実現できる。

*3 某動画サイトでこういう発言をすると、「作者は病気」のタグが頂けるらしい。

書籍紹介

最後の章では、WinAPIについて触れているので、Windowsプログラミング未経験者でも1から学ぶことが出来ます。こちらの記事は、[登大遊](#)^{*4}さんが執筆されているので、登さんだいすき！な方にもオススメです。

書籍名：【書籍版】DirectX9 実践プログラミング

著者名：大川善邦

成田拓郎

大澤文考

登 大遊

発 行：工学社

ISBN-13：978-4875934998

値 段：2940 円

発行日：2003/9



^{*4} 魔法のトンネル PacketiX VPN の開発者、筑波大学発のベンチャー企業 SoftEther（株）の会長さんでもある。実は WORD 部員。

情報科学類誌

WORD

WORD は HD-DVD を応援します号

発行者

情報科学類長

編集長

武井 裕也

制作・編集

筑波大学情報学群

情報科学類誌WORD編集部
(第三エリアC棟212室)

印刷

総合研究棟B棟印刷室

2008年1月15日

初版第一刷発行(512部)