

WORD

From Colledge of Information Science 2009.11

◆祝!WORD30周年!

- ・WORD30周年
記念イベントレポート
- ・WORD Memories

12

◆カメラ好き必見!

- ・GRな日々。

◆台風好き必見!

- ・台風と戯れる Zwei!

◆情報系記事も充実!

- ・文明の進歩の
尺度への道
- ・ICT中間報告会
- ・入門 Ruby on Rails

◆感動の超大作!

- ・長編大作
「男の娘と箱ガンダムと」



WORDは
一太郎を使っています号

WORDは
一太郎を
使っています号 目次



-
- WORD Memories P. 3
-
- WORD30周年記念イベントレポート P.18
-
- ICT中間報告会 P.27
-
- 文明の進歩の尺度への道 P.31
-
- GRな日々。 P.43
-
- 入門 Ruby on Rails P.46
-
- 台風と戯れる Zwei ! P.52
-
- 長編大作「男の娘と箱ガンダムと」 P.56

WORD Memories

WORD 30th Anniversary

文 編集部 いのひろ

WORD創刊30周年

WORD、30周年おめでとう！

1979年1月26日、我らが情報（科）学類誌WORDが産声をあげました。絶滅していく学類誌が多い中、偉大な先輩方のご苦労により、30年もの間絶えることなく発行され続けてきたWORDですが、その間には様々な出来事があったようです。本記事では筑波大学 / WORD年表と共に、WORDとWORDに携わってきた人々の軌跡を振り返ってみたいと思います*1。

本記事は2009年9月26日に行われた「WORD創刊30周年記念イベント」で配布された、「30周年記念総集編」に収録されたものを、通常号向けに再編集したものです。

1. 筑波大学開学～情報学類誌WORDの誕生

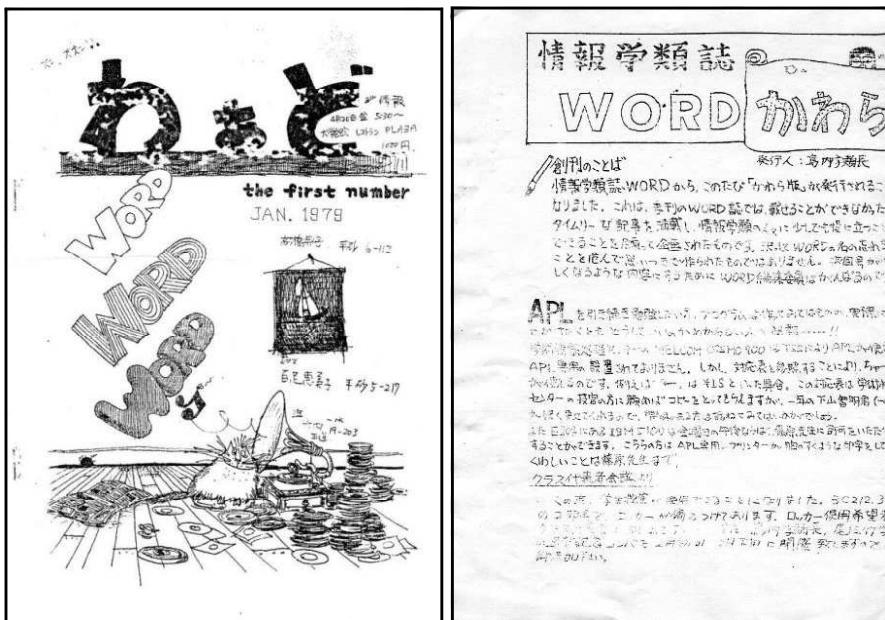
W O R D 時 代	1973	筑波大学が開学。 第一学群、医学専門学群、体育専門学群、附属図書館設置。	1973年、筑波大学が開学した。開学時は第一学群、体育専門学群、芸術専門学群の3つの学群しか存在せず、情報学類が含まれる第三学群が設置されたのは開学から4年後の1977年のことであった。
	1975	第二学群設置。 芸術専門学群設置。	この頃、多くの学類が学類誌を創刊・発行していた。その中に、情報学類誌WORDも存在していた。
	1976	筑波大学医学専門学群附属病院開院。	WORDの創刊は1979年1月26日。学生間、および学生と先生との情報交換に役立ちたいという願いのもと創刊された。編集長は中尾敏行氏。
	1977	第三学群設置。 (情報学類を含む)	
黎	1979	情報学類誌WORD創刊。 (1979年度は第5号まで発行された)	創刊号には現在も筑波大学で教鞭を執ってらっしゃる油田信一先生（現在、工学システム学類）や、筑波大学名誉教授であり元情報学類長であった森亮一*2先生のお言葉が寄せられている。以下は、森亮一先生の記事「無限に近い可能性について」からの抜粋。
	1980	発行号数のふり直し？	

*1 本記事は2007年2月発行「さよなら情報学類号」の「WORD Memories（編集者：K TOK）」をもとに大幅に拡張を加えたものです。K TOK先輩に感謝。

*2 筑波大学名誉教授。元情報学類長。独自のデジタル・コンテンツの流通システム「超流通」を考案。1986年、IFIP（国際情報処理連合）からSilverCore賞を受賞。

WORD 史（通常号版）

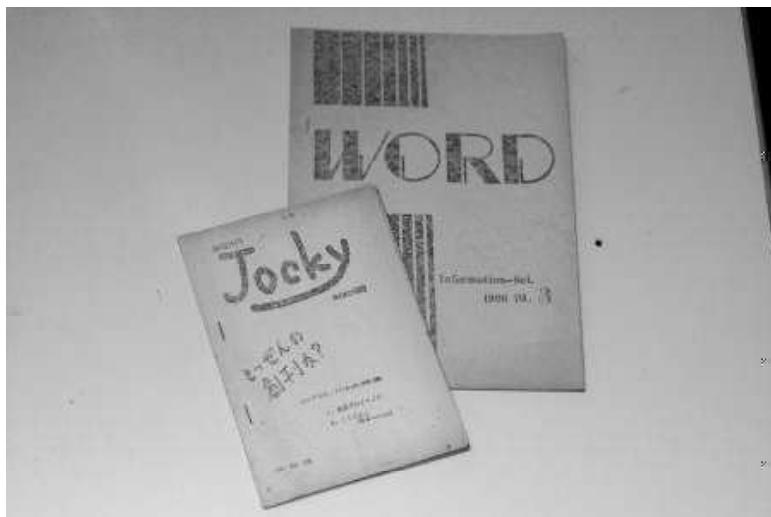
明 期	かわら版の発行. 国際関係学類設置。 (この年までに 10 号が発行された) 日本語・日本文化学類設置。 WORD 別冊「Jocky」が発行されるものの短命に終わる別の別冊として「extra」が発行される。	“最近、8086 等の性能で十分であり、その先に進んでも何に使うのかという意見を聞くことがある。これは全く誤っている。我々は、大きな山に登った のではなく、山の麓にたどりついた処なのであって、マイクロコンピュータには無限に近い可能性が残っているのである。” 略) “一台の計算機はプログラムを実行するという点でやつと、1 個の遺伝子と比較できるかもしれないが、その上には細胞、組織、器管、個体、異なる種からなる社会、宇宙と続くのである。我々は、大きな山に登った のではなく、麓にたどりついたに過ぎない。”
--------	---	--



WORD 創刊号表紙と WORD かわら版

1980 年 1 月には「WORD かわら版」が創刊された。かわら版には季刊の WORD に掲載できなかつたタイムリーなネタを随時掲載していたようである。

1986 年には「WORD 別冊 Jocky」が創刊された。WORD が B5 版であるのに対して、Jocky は B6 版で印刷された。Jocky の内容は「広辞苑[筑波版]」「筑波グルメマップ」「高速バス時刻表」など手軽にどこでも持つて行ける便利な WORD という位置づけで作られたようである。



1986-03 「Vol.3」と「別冊 Jocky 創刊号」の比較

創刊時からしばらくは「WORD 編集委員会」という団体名で活動しており、学類直属の団体であったことがうかがい知れる。なおあまり知られていないが、「WORD」という名称は Microsoft Word^{*1}ではなく、情報量の単位である「word」に由来している。

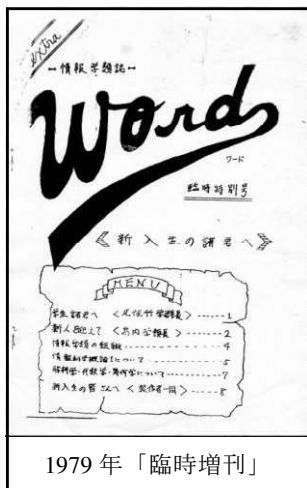
この時代の秀逸な記事

- ・ 1979-01 創刊号「無限に近い可能性について」（森亮一先生）
 - ・ 先生方のコンピュータサイエンスにかける熱い思いが伝わってくる。感動
- ・ 1979-01 創刊号
 - 「学生時代を振り返る -僕は学生の時何をやっていたか-」（油田信一先生）
 - ・ 油田先生が学生時代、どのように遊んだかという記事
 - ・ 車 10台くらいでよく火力発電所や高萩の KDD 通信基地などに行ったりらしい
- ・ 1985-06 VOL.1 「安くて早い国鉄を使った帰省の方法—概論—」
 - ・ 国鉄を使って帰省するときの、お得なキップなどを紹介する記事
 - ・ 在りし日の国鉄。東海道線の東京一大垣直通車がある。
 - ・ 18きっぷが 1回 2000円、新幹線にのぞみがないところからも歴史を感じる
- ・ 1986-05 DATABASE 1-2
 - ・ 当時の情報学類生に対して行ったアンケートをまとめた冊子
 - ・ 23年も前のアンケートにもかかわらず、現代と同じようなことが書いてある
- ・ 1986-? VOL3. 「私の苦手なもの」
 - ・ 情報学類生の日常を漫画で描いた記事
 - ・ 内容も、共感できるところが多いかもしれない

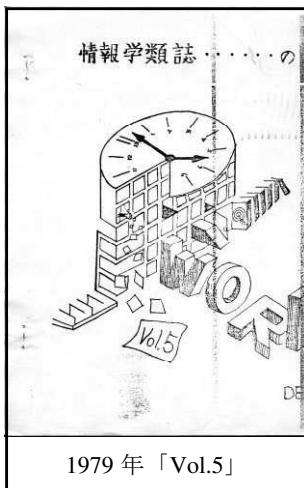
*1 ちなみに Microsoft Word の初版は 1983 年に発売された。

WORD 史（通常号版）

この時代に発行された WORD の表紙



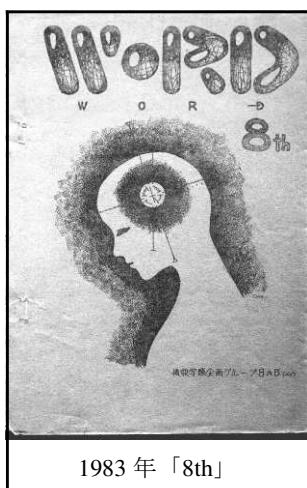
1979年「臨時増刊」



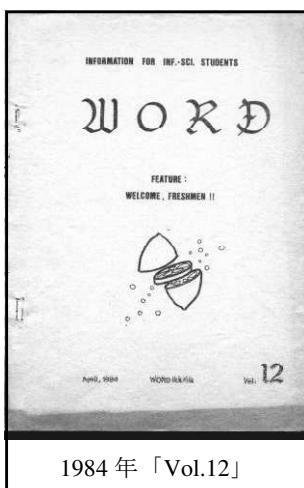
1979年「Vol.5」



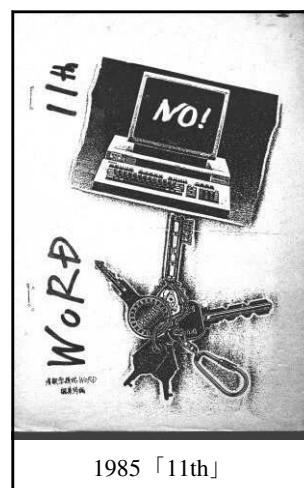
1980年「Vol.4」



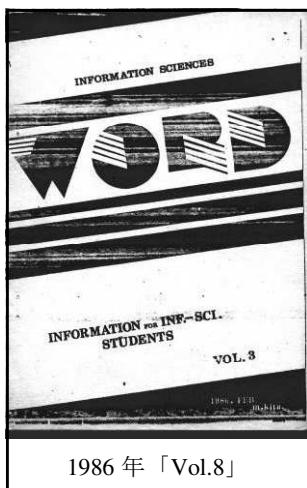
1983年「8th」



1984年「Vol.12」



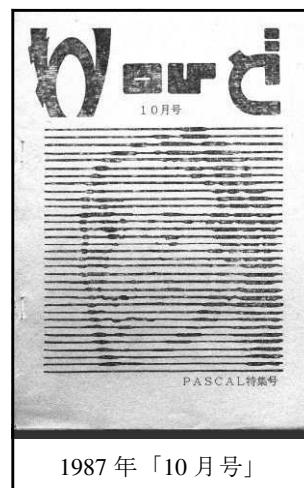
1985「11th」



1986年「Vol.8」



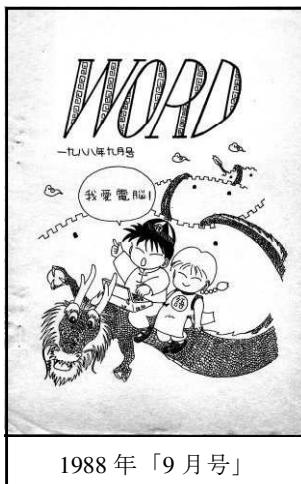
1987年「6月号」



1987年「10月号」



1988 年「5・6 月合併号」



1988 年「9 月号」



1988 年「12 月号」

2. 暗黒時代

暗 黒 時 代	1989	「WORD 編集委員会」が「WORD 編集部」に改名？ 36 号（推定）を最後に累計発行号数のカウントがなくなる。
	1990	80 年代後半から 90 年代の初頭にかけて手書きで絵が描かれた表紙の WORD が多く発行された。
	1994	情報学類計算機が HP に。

1980 年代後半から 90 年代中盤まで、各学類誌は暗黒時代を迎えた。数多く存在した学類誌の大半がこの時代に姿を消し、残ったのは WORD を含む幾つかの学類誌のみであった。WORD 以外のものを挙げると、社会学類誌「そおしあ～る」、農林学類（現在の生物資源学類）誌「の一のりんりん」などがある。

「WORD 編集委員会」から「WORD 編集部」へと名称が変更されたのは、暗黒時代のまっただ中の 1989 年であると考えられている。また創刊時には第三学群 A 棟脇（現在の学生控室）にあったはずの WORD 編集室は、この時代の間に C 棟 213 号室（現在の情報科学類学生ラウンジ）に移動したと考えられている。

この時代の秀逸な記事

- 1991-? 秋季号「筑波今昔物語」
 - ・ 筑波研究学園都市と筑波大学創設の頃の話をまとめた記事
 - ・ 昔の「筑波送り」は本当に「島流し」と同じようなレベルだったようだ
- 1991-? 秋季号「他学類から見た情報学類」
 - ・ 他学類に、情報の人のイメージについてアンケートを採ってまとめた記事
 - ・ 当時から情報学類のイメージは変わらないようである
- 1992-10 '92 秋季号「女の子のページ」
 - ・ 当時の情報学類に所属している女の子に対話形式でアンケートをとった記事
 - ・ (今となっては)珍しい女性陣による対談記事。雰囲気が全然違う
- 1993- 秋季号「筑波車壊学基礎論」
 - ・ 筑波での免許の取り方（今では古い）と、車と人間と警察と鳥のフンが対決したらどうなるか？などの考察記事
 - ・ 車がないと遊び的な意味での生活が苦しいのは 25 年前も同じようである

WORD 史（通常号版）

この時代に発行された WORD の表紙



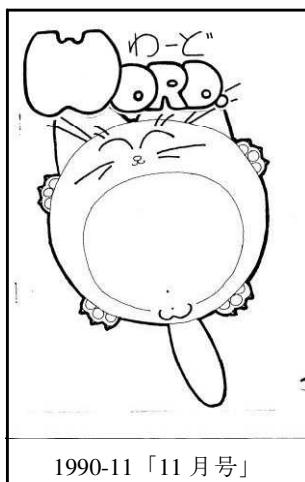
1989年「6月号」



1989-10「バグ慰靈祭特集号」



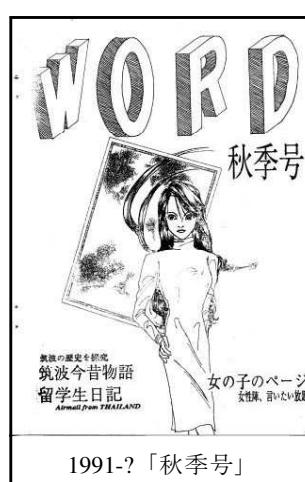
1990-05「五月号」



1990-11「11月号」



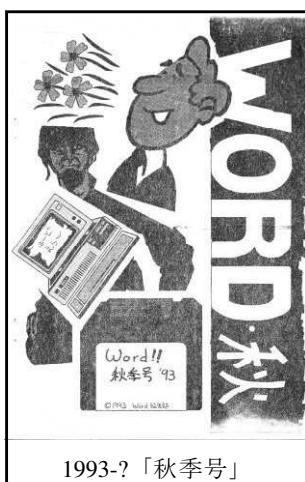
1991-?「4年生追い出し号」



1991-?「秋季号」



1992-10「10月号」



1993-?「秋季号」



1995-?「秋季号」

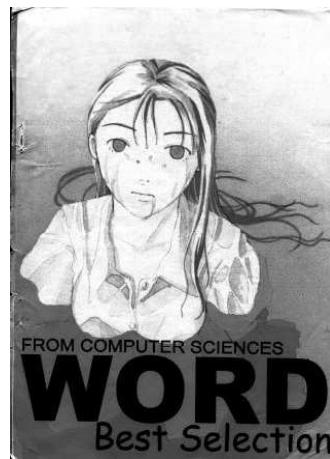
3. 戦国時代

戦 国 時 代	1996	新歓の乱が勃発。 WORD 編集部と新入生歓迎委員会が分裂。
	1997	この頃から「Just Systems 一太郎」を用いて記事を書くようになったと考えられている。
	1998	情報学類計算機が SGI に。新歓号で「Q&A」などの名物記事（現在も使われている）が生まれる。
	1999	棍棒事件。
	2000	筑波大学が AC 入試を開始。「Best Selection」として某イベントで配布。
	2001	編集部の三権分立が確立。編集室が現在の位置に。

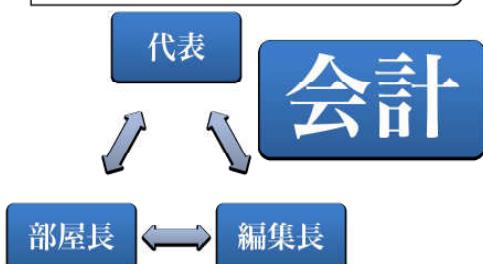
WORD 編集部はかつて、情報学類の学生行事（新歓^{*1}、バグ祭^{*2}など）を取り仕切っていたが、1996 年に新歓の乱が勃発する。これにより、WORD 編集部と新入生歓迎委員会が並立する現在の体制が発足した。

また 1996 年以降、編集部内では勢力を拡大した一部の部員による統治がしばらく続くことになった。この時代の有力者として、1997 年度入学生の長田敏之氏が挙げられる。彼は精力的な活動によって今日の WORD の基礎を作ったとされており、現在でも編集部内で高い評価を得ている。

2000 年には某世界最大の同人誌即売会に「より抜き WORD (Best Selection)」を引っさげて参加。表紙はカラー印刷であった。



WORD 編集部における三権分立



2001 年、編集室が現在の C 棟 212 室に固定された。部屋面積がそれまでの 3 分の 1 となったが、交換条件として電力及びネットワークの供給を受けられるようになった。また、同年に代表、編集長、部屋長の三権分立体制^{*3}が確立され、WORD 編集部は君主制から民主制へと移行した。

*1 WORD 編集部における新歓は WORD 編集部が行っている。

*2 かつてバグ祭は情報学類の一大イベントと言えるものであったが、現在ではちょっとした内輪の飲み会程度の規模になっている。

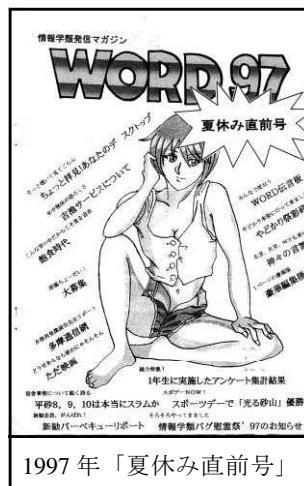
*3 実際には会計が最高権力者であることは言うまでもない。

WORD 史（通常号版）

この時代の秀逸な記事

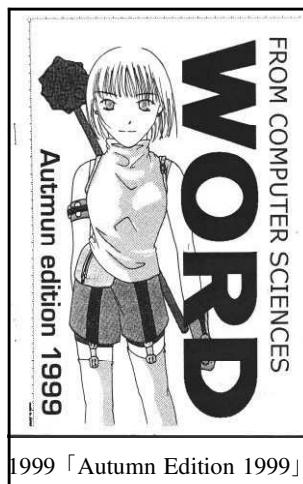
- ・1997-12 雪だるま号「Load Test」
 - ・色々なモノと間違えて CD を焼いたり切ったりする小説？記事
 - ・「しまった、これは CD-ROM だ！」
- ・1998-10 秋季号「ゲーム研究隊、北へ！-特別編- 副学長と打つ」
 - ・当時の副学長と麻雀をする記事
 - ・副学長と交流があったことに驚かされる
- ・1999-06 20周年記念号「Windows98 は二度死ぬふたたび」
 - ・これより前にかかれた「Windows98 は二度死ぬ」という記事がどこかの学類で授業に取り上げられ、さらには期末テストにも取り上げられたらしい。その珍解答とコメントが掲載されている
 - ・WORD の記事が授業で取り上げられたというのがすごい
- ・1999-11 秋期号「タイ日記」
 - ・タイへ旅行(修行？)に行った時の日記。向こうでの生活の苦労などがかれている
 - ・タイコラムが良い。国による価値観などの違いを考えられる良記事
- ・2000-11 秋季号「考察」
 - ・童謡「森のくまさん」の歌詞を、これでもかと言うほどに隅々まで「考察」する記事
 - ・屁理屈は情報学類生のお得意技だっ！てか？
- ・2001-? さよなら O2 号「とんでも広告の世界」
 - ・パワーストーンなどの広告を取り上げその文面に突っ込みを入れる記事
 - ・美辞麗句を並び立てても、商品の怪しさはむしろ際だつというものである

この時代に発行された WORD の表紙





1999 年「20 周年記念号」



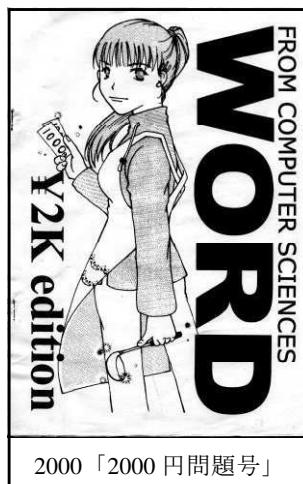
1999 「Autumn Edition 1999」



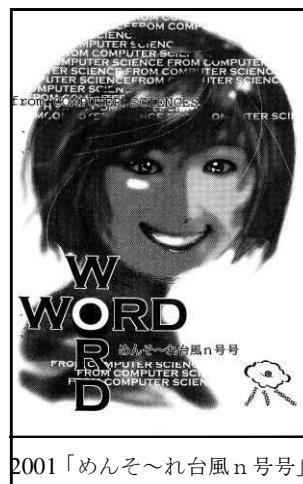
1999-? 「かたつむり号」



2001 「21st Autumn Edition」



2000 「2000 円問題号」



2001 「めんそ～れ台風n号号」

4. AC 時代の到来

A C 時 代	2002 編集部内での AC の台頭が始まる。 情報学類計算機が DELL に。 筑波大学と図書館情報大学が統合、図書館情報専門学群設置。 医学類、看護・医療科学類設置。	筑波大学は 2000 年度入学試験から AC 入試を実施し、専門分野において高い問題解決能力を持つ学生の獲得に力を入れてきた。一方、AC 入試以外の試験で入学してきた学生、あるいは AC 入試実施以前の学生にも AC 入学生に引けをとらない能力を持つ学生が存在した。いつしか WORD 編集部はそういういた学生が集まるような場所になり、その流れは 2002 年以降いっそう顕著になった。
	2003 編集部が情報学類視察団の査察を受ける。 工学システム学類生が編集部に加入。	編集部内での「AC」という言葉の意味は本来のものから拡張され、「高度な専門技術・能力を持つ人間。あるいはそういういた人間が行うような、傍から見たら奇妙とも思える行動」というニュアンスへと変化していった。そしてこの新しい意味での「AC」は、技術者や研究者を通じて大学外まで広がっていくことになる。

WORD 史（通常号版）

2004	国立大学法人筑波大学発足。 ソフトイーサ株式会社設立。 女性が初めて WORD の代表になる。 工科生が初めて WORD の代表になる。	2004 年、WORD 編集部員の一人であった登大遊氏を中心とした学生・院生により、ソフトイーサ株式会社が筑波大学発ベンチャー企業として設立された。ここに、編集部における「AC」のヒエラルキーは頂点を迎えることとなった。
------	---	--

この時代の秀逸な記事

- ・ 2002-? 図情のもつれ号
「プロジェクト W 一編集者たちー 自販機を空にした男たち」
 - ・ 何故か学祭当日に自販機を空にしようと立ち上がった男達の物語
- ・ 2002-? 前の後の祭り号「神々の言霊」
 - ・ 情報学類の神々の発言をまとめたシリーズ記事
 - ・ 名言（迷言？）を発する神はだいたい特定の先生ではないだろうか
- ・ 2003-? UT2003 号「ProjectW センター試験に挑んだ男、バグを慰靈した夜」
 - ・ AC 入学生などがセンター試験を受験してみる記事
 - ・ WORD のノリは今も昔も変わらない
- ・ 2003-? 「Hello, Word!」（連載）
 - ・マイナーなプログラミング言語の紹介をする連載記事
- ・ 2004-01 SP2 対策号「ハイブリッド電子工作」
 - ・ まじめな電子工作記事からシームレスに話題がすり替わっている怪作
- ・ 2004-? がんばれ一太郎・花子号「AC の法則」
 - ・ WORD における AC 系記事の最高峰
 - ・ 「遅くとも 13 歳位までの間に誤ってまたは故意にコンセントなどに触って感電したことのある人は AC になる」などの法則が生まれた

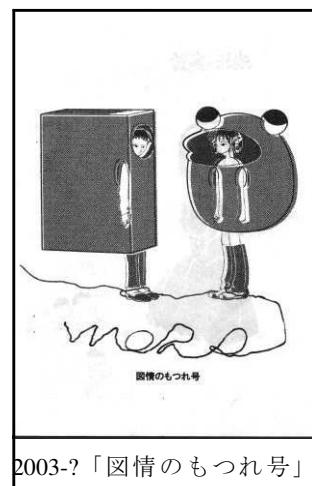
この時代に発行された WORD の表紙



2002 「FIFO ワードカップ号」



2003 「学園祭直前号」



2003-? 「図情のもつれ号」

5. 学群再編、「情報学類誌 WORD」の終焉

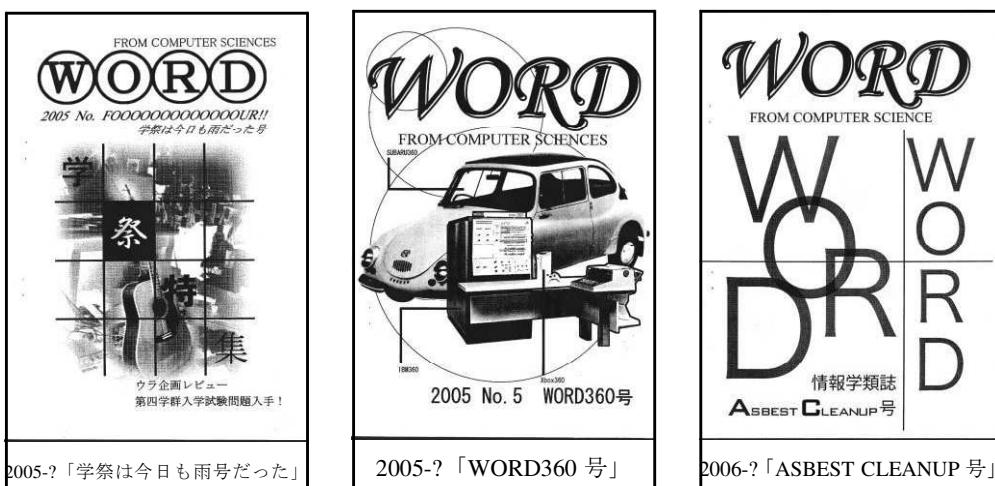
激動期	2005	学群再編の発表。 「WORD Best Selection3」を某イベントで配布。 つくばエクスプレス開通。	2005年8月、つくばエクスプレスが開通。つくばと秋葉原を最短45分で結ぶ路線の誕生によって、つくば市が、筑波大生が沸いた。
	2006	情報学類計算機がAppleに。学群再編の詳細が決定。 某SNSに「WORDが嫌い」コミュニティが登場（編集部員が続々と参加）。 情報学類誌WORD消滅。	その一方で、筑波大学とWORD編集部は激動の時代を迎えることになる。2005年7月、学群・学類再編が公式に発表された。その後、2006年3月に改組後の学群・学類組織が正式決定した。これにより、情報学類を第三学群から独立させ、図書館情報専門学群と統合させた「情報学群」という新しい学群を作ることが決定した。

これを受けて編集部内では「ついに『学群誌』になるのか」という気勢が高まった。一方で、「どさくさに紛れてWORDが廃刊になるのではないか」という心配をする声もあった。学類長との話し合いの結果、WORDは「情報科学類誌」として再編後も存続することになった。

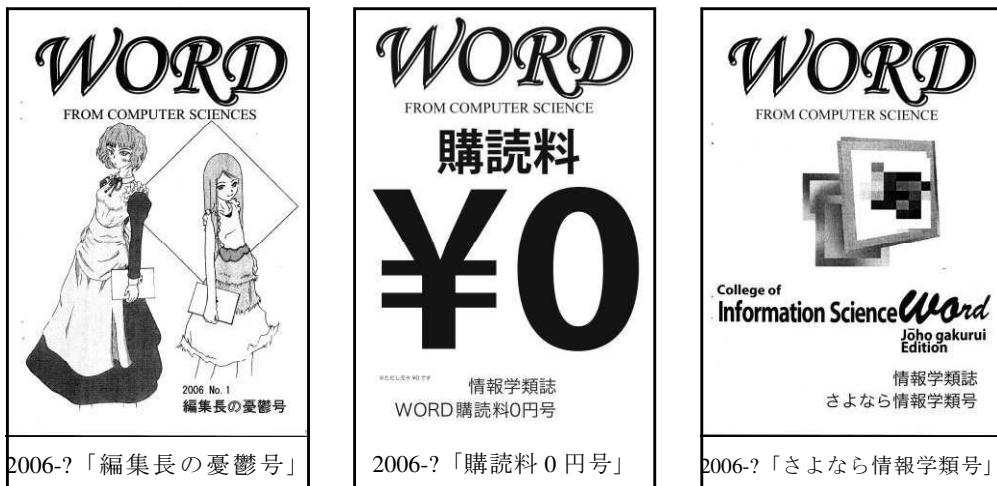
この時代の秀逸な記事

- ・ 2005-? WORDEXPO号「うまい棒早食い大会 in WORD編集部」
 - ・ 1時間ひたすら「うまい棒」を食べ続けた男たちの話
 - ・ 6人で1時間、1人あたりの平均44本
- ・ 2005-? TX男号「頭文字W」
 - ・ 自転車で伝説の公道バトル「ループ1周」
 - ・ 最下位の罰ゲームは「3日間自転車ロック」。筑波大学にとってはまさに地獄
- ・ 2006-? さよなら情報学類号「WORD史」
 - ・ WORDの生い立ちが紹介された記事
- ・ 2006-? 「フリーゲームのすすめ（レクチャー&レビュー）」（連載）
 - ・ フリーゲーム制作ツールの使い方、有名フリーゲームなどのレビュー記事

この時代に発行されたWORDの表紙



WORD 史（通常号版）



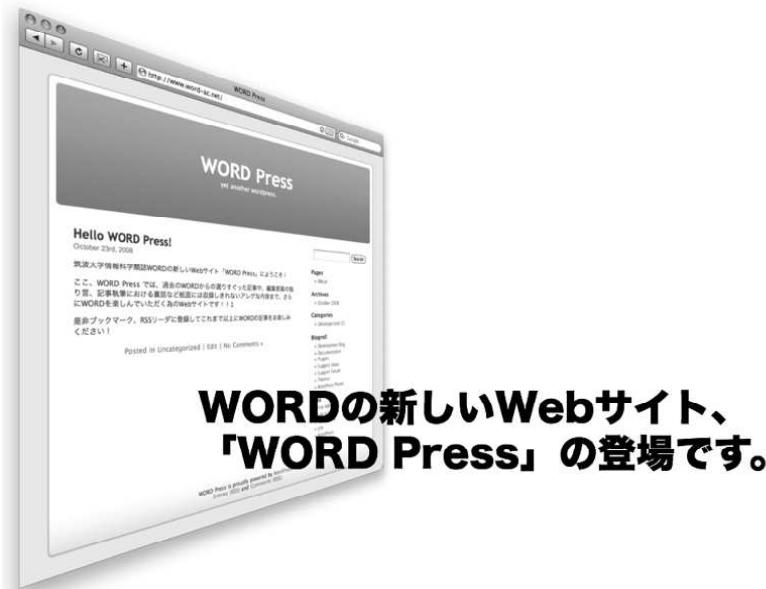
同じデザインの WORD ロゴが使われるようになり、近年の WORD の表紙に近いレイアウトになってきた。号の名称もこれまでの季節に関するものなどが多かったものから、時事ネタが使われるよう変化しているのがわかる。

6. 「情報科学類誌 WORD」の誕生～現在

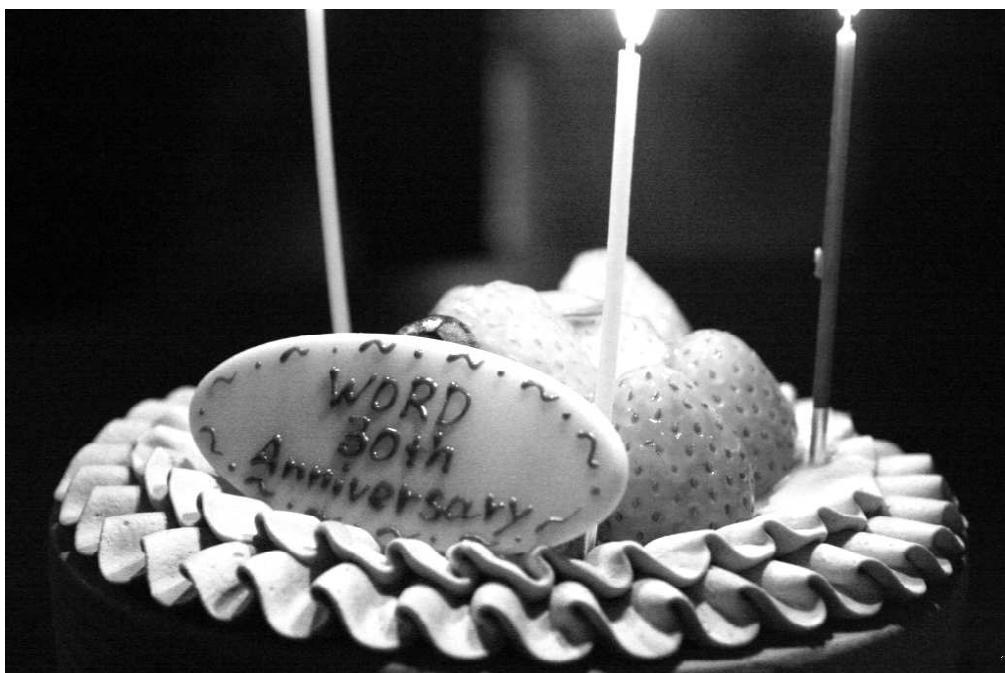
近 代 期	2007	学群・学類再編施行。 情報学群・情報科学類設置。 情報科学類誌 WORD 誕生。 現代思想特講Ⅱ開講。 ポテト祭,Project MAX 実行。	2007 年 4 月より学群・学類再編が施行され、情報学群情報科学類が発足した。WORD は情報科学類誌として新たなスタートを切った。 しかし新たな WORD の門出を飾るべく制作した第 1 号であるが、学類長の発行許可が下りず修正命令が下された。「発禁（発行禁止）」を食らったのはこれが WORD 史上初であると言われている。
	2008	MAX プリンを持って新歓に凸（Star Words）。 WORD Press 開設。 3A 棟改修工事完了。 雙峰祭で MAX プリンを販売。	2007 年 10 月頃、MAX コーヒー中毒者による企画が突然発し、後に伝説となった「MAX コーヒープリン」が誕生する。
	2009	WORD 創刊 30 周年。 3B 棟改修工事完了。	2008 年 4 月には編集室の荷物を一度すべて 3C213 ラウンジに派出し、カーペットを敷くという大掃除プロジェクトを完工。匠の技によって居住性を高めることに成功した。

2008 年度の新入生が大量に押し寄せた。引っ越し準備号・新入生歓迎号が好評だったのが要因らしい。しかし最終的に常駐するのは 3 人に落ち着いた。

2008年10月には新しいWORDのWebサイトとして「WORD Press」を設置^{*1}。第1号からのバックナンバーをPDFで公開している。ちなみに日本で31番目にIPv6に対応したWebサイトである。



2009年1月26日、ついにWORDが創刊30周年を迎えた。



*1 <http://www.word-ac.net/>。ブログエンジンには当然Wordpressを採用。

WORD 史（通常号版）

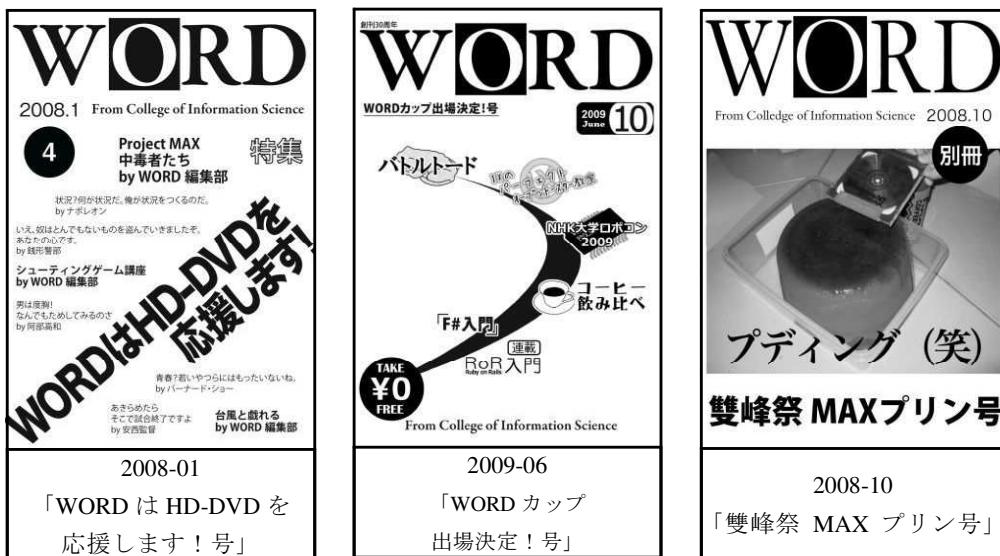
この時代の秀逸な記事

- ・ 2007-06 Hello WORD 号 「ロリ狐。」
 - ・ Firefox のロリカスタマイズ版である「lolifox」の紹介記事
 - ・ WORD 史上初めて発行禁止処分を受ける。「ロリ」は特に問題ではなかったようだ
- ・ 2007-09 消えた A 棟の謎号「ようこそ☆ WorDonald へ」
 - ・ フライドポテト祭のまとめ記事。もとい、ポテト災
 - ・ ノルマは 1kg/人であった
- ・ 2007-11 真ゆとり世代。号「ヲタ的雙峰祭」
 - ・ WORD ヲタ記事の最高峰
 - ・ モモーイさんへの愛に溢れる記事
- ・ 2008-01 WORD は HD-DVD を応援します！号「Project MAX -中毒者たち-」
 - ・ MAX コーヒーを利用した料理記事
 - ・ MAX コーヒープリンばかりが有名になってしまったが、鍋やぜんざい等も製作された過去があることはあまり知られていない…¹
- ・ 2009-05 WORD で何が悪い！号「つくばデカ盛り名鑑」
 - ・ 大学周辺のデカ盛り料理を出す店を紹介
 - ・ 筑波大生の食生活には欠かせないもの、それは「重食」である

この時代に発行された WORD の表紙



*1 というかあまり思い出したく(略)



2007 年に情報科学類誌 WORD になってから、表紙のレイアウトはほぼ固定された。また号番号は年度に関係なく、通し番号になった。

最後に

簡単にではありますが、WORD の軌跡を振り返ってみました。いかがでしたでしょうか。

他の学類誌が創刊され、そして廃刊してゆく中、WORD が 30 年もの間発刊し続けられたのは、自由さがあったからではないでしょうか。時にはお叱りを受け、時には発禁を受け。様々なことがありました。この自由な雑誌が書き手からも読み手からも愛されていたからここまで続いてきたのでしょう。

これからも適当に自由に、ほどほどに奔放に。お叱りを受けないようにそれなりに続けて行きたいと思いますので、是非みなさんも期待し過ぎずに WORD と付き合ってやって下さい。

レポート：WORD創刊30周年記念イベント

文 編集部 いのひろ

2009 年 9 月 26 日（土）に一学食堂にて、「WORD 創刊 30 周年記念イベント」を行いました。本記事ではそのイベントの様子をお伝えできればと思います。

2009年9月26日（土）一学食堂

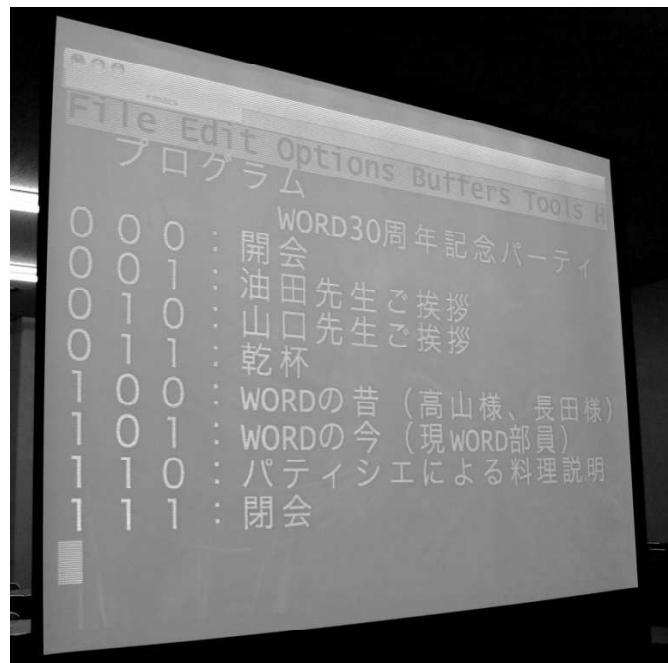
読者の皆様のおかげで、WORD は今年創刊 30 周年を迎ました（しつこい）。9 月 26 日に行われた記念イベントでは、OB・OG、WORD 編集部にゆかりのある先生方、在学生を招待し、まるで成長していない WORD について懇談しました。



ここで一つお詫びがあります。読者の皆様への本イベントの詳細な案内は、紙面上で行いたいと考えていましたが、配布時期の都合が悪く、それができませんでした。このためビラを作製し配布しようと考えていましたが総集編の編集が大炎上していた為、配布するのを忘れてしまっていました。まさか WORD 創刊 30 周年のイベントを楽しみにしていたというような奇特な方がいらっしゃいましたら、心からお詫び申し上げます。WORD 編集部（3C212）まで来ていただければ、このイベントでしか配布を行わなかった「記念総集編」を差し上げます。

さて気を取り直して、本題へ。この写真はプロジェクトに映し出されたイベントのプログラムです*¹。

*1emacs に記述していますが、WORD 編集部は vim 派が多いです



まず開会として創刊号に寄稿いただいた、現在は工学システムで教鞭を執っていらっしゃる油田信一先生にご挨拶をいただきました。



「情報科学類誌 WORD」という名称は、創刊当時に刊行されていた「bit 誌」と「BYTE 誌」が由来となっていることや、廃止されていく学類誌が多いなか生き残っていること、これからも続いて行って欲しいというようなお言葉をいただきました。

レポート：WORD 創刊 30 周年記念イベント

続いて、WORD の発行者として日頃から WORD がお世話になっている、情報科学類長 山口喜教先生からご挨拶をいただきました。現在 WORD 編集部では過去 30 年分の記事をすべてデジタルアーカイブする計画がありますが、学類としても協力していただけるということです。



さらに WORD のイベント当時の編集長である yasuharu が挨拶をしました。



その後に司会者より祝電や記念品などを紹介がありました。1986 年入学の方からいただいた電報などが紹介され、会場の壁に掲示されました。

レポート：WORD 創刊 30 周年記念イベント

乾杯のご挨拶は加藤和彦先生にいただきました。加藤先生は筑波大学情報学類ご出身で、在学中は WORD 編集部員では無かったものの、毎年開かれていた「プログラムバグ慰靈祭」の準備などを行ったそうです。思い出話ををしていただきました。



乾杯の後は適当に歓談。過去の WORD を持ち寄った OB の方がいらっしゃり、昔話に花が咲いていました。

適当に時間が経ったところで、プロジェクタを用いてプレゼンタイム。まずは 1997 年度入学の長田さんです。長田さんは 2000 年度前後に WORD 編集部で精力的に活動され、様々な種類の記事を執筆されました。今回は長田さんが在籍された 2000 年前後における編集室の変遷を紹介してくださいました。



レポート：WORD 創刊 30 周年記念イベント



また秀逸な記事として総集編に再収録された「副学長と打つ*」の取材で撮影された写真も公開されました。

*1 1998 年度秋季号（1998 年 10 月発行）に収録された「ゲーム研究隊、北へ！ -特別編- 副学長と打つ」。その名の通り、当時の副学長と麻雀を打った記事



続いて、高山さんから当時の様子のお話をいただきました。高山さんは情報学類の第5期生で、WORDの初期をご存じでした。プログラムバグ慰靈祭では、「**学年で一番かわいい男の子に御子服を着せ、ひたすら樽酒を飲む**」のが本来のあり方だったそうです。



レポート：WORD 創刊 30 周年記念イベント

次に近年の WORD 編集部の紹介として、現編集部員の m-bird が紹介を行いました。現在の編集室の様子、編集の過程、女性編集員の紹介などが行われました。



最後に「パーティシェによるデザート説明」が行われました。勘の良い読者なら既に気づいているかもしれません、私のひろが MAX プリンについて発表いたしました。



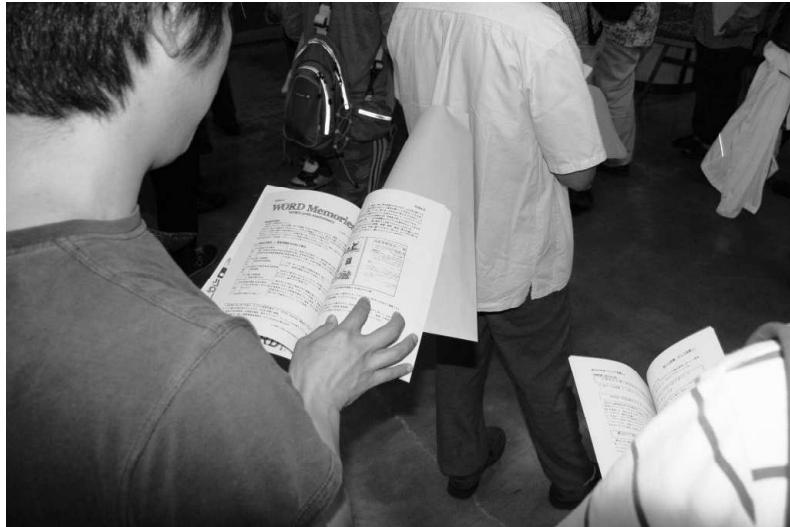
発表の最後には実際に 8 リットルバケツプリンが登場し、振る舞われました。思いのほか好評で安心しました。

最後に今後の WORD と WORD 編集部の発展を祈って一丁締めを行い閉会となりました。受付では 30 周年記念総集編が配布されました。

30 周年記念創刊号はまさかの 128 ページ構成になってしまっており、編集部員で夏休みの終わりから保存されているすべての WORD に目を通し、秀逸な記事を選びました。

WORD 史 (WORD Memories) は若干変更が加えられていますが、本号に掲載されているので是非読んでみてください。

レポート：WORD 創刊 30 周年記念イベント



以上、9月26日に行われたWORD創刊30周年記念イベントの様子をまとめました。少しでも雰囲気が伝われば幸いです。30年というキリの悪い数字でイベントを開いてしまったことに少々後悔しつつ、創刊32周年記念イベントの開催予告は断言できないまま終わらせていただきます。

今後もWORDと、WORD編集部をよろしくお願ひいたします。

ICT 中間報告会

文 編集部 ミレトス

はじめに

皆さん、ICT とは何の略がご存じですか？

ご存知、ないのですか！？^{*1}

ICT とは Information and Communication Technology の略で、日本語で言うと情報通信技術を表す言葉です。そして今回皆さんにお伝えするのは、コンピュータサイエンス専攻^{*2}で実施されている「ICT ソリューションアーキテクト育成プログラムの中間報告会」の様子です。

この中間報告会は 10 月 11 日、つまり**雙峰祭の日**に開催されていました。場所はどこかというと、総合交流会館です。**筑波大学って広いよね！**

本編

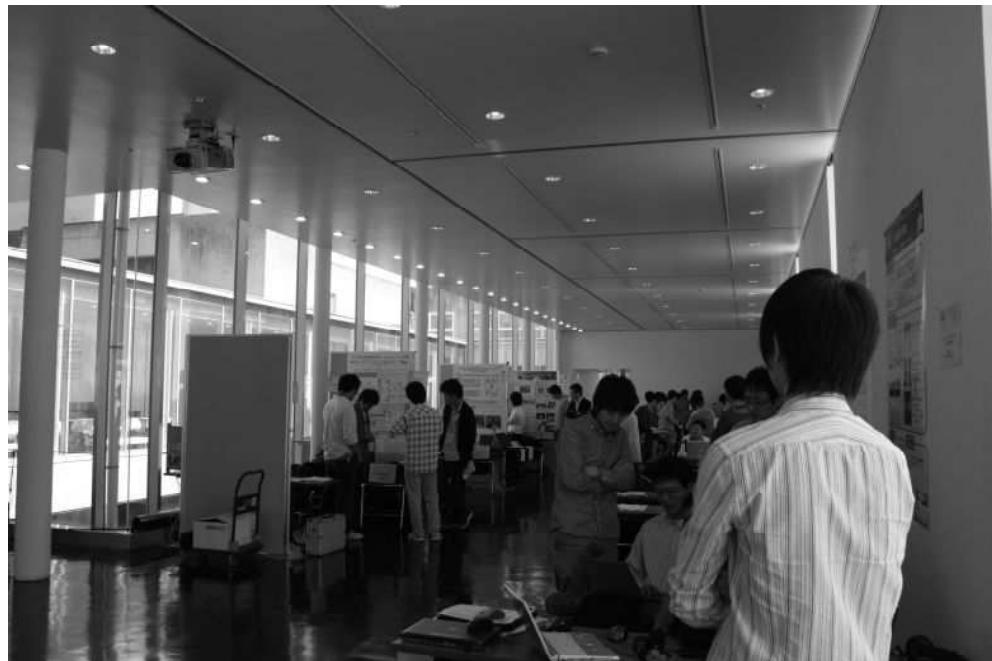
それでは早速中間報告会の様子をお伝えていきましょう。



会場となった総合交流会館の前日の様子です。実は大学会館と繋がっている場所なのですが、筆者はどこだかさっぱりわかりませんでした。中に入ってみると日の入りがよく、開放的な感じがしました。

*1 マクロス F の名ゼリフ

*2 大学院での専攻の 1 つ



当日の会場の様子です。多くの人が来ており、大学院の方々が自分達のプロジェクトの説明をしていました。その中で特に興味を引かれたもの（＝筆者の頭でも理解できるもの）を紹介していきたいと思います。

・賑やかさを伝えるライブカメラコミュニケーション

まず初めに興味を引かれたのは画面内で飛びまくっている花火です。これはどういうものか、というのを簡単に説明すると、マイクのある場所の賑やかさを花火で表してみよう、というものです。離れた所からその場の映像を見ている人が、その場の賑やかさを把握するために使ったりするそうです。それならマイクの音声情報をそのまま送ればいいんじゃないの？と質問したところ、それではプライバシーの問題が発生する、ということでした。花火は全部で5種類あり、賑やかになればなるほど花火が大きくなり、たくさん打ち上がる仕組みになっています。



花火の強さの度合いです。表示5が一番大きな花火です。

・自動生成と修正インターフェースによる靴型設計支援システムの開発

靴は、まず最初に靴型を作成し、それに合うように作っていきます。ここでは、その靴型を作る手助けとなるシステムを紹介していました。



これが靴型です。今はまだ職人による手作業で作られているらしく、触ってみるとすごくすべすべとして気持ちよかったです。

・DOCoCa：行方履歴を用いてメンバーの習慣を可視化する電子行方表

最初に見たときは果てしなくグレーディングに近い何かを見てしまったと思いましたが、実際は全く別の内容で安心しました。行方表とは、「今どこにいるか」という情報を他の人に教えるための表で、オフィスなどでよく使われています。これを使って、メンバーの習慣を他の人が把握する為のシステムを作っているということです。これによってお互いをより理解したコミュニケーションを図る事が出来るようになりますのが狙いだそうです。とてもユニークな表示の仕方をしており、使っている人の習慣、いつからどこにいるのかという情報が見ただけでわかるようになっています。



DOCoCa の様子です。画像では見づらいですが、渦巻き状に表示しており、中に行けばいくほど古い情報、外側が新しい情報となっています。

・マルチマウス・マルチディスプレイによる統合操作環境

これは、複数のディスプレイの中を複数のポインタが動き回っているという面白い画面に惹かれました。同じパソコンにディスプレイ 2 台、などというのはよく見かけますが、これは別々のパソコンのディスプレイの中をポインタが移動していました。ウィンドウも別々のパソコンのディスプレイをまたいで表示するし、技術とかさっぱりの筆者から見ると、これって意外と凄い事なんじゃないか？と思わせるような内容でした。ディスプレイの様子伝えられないのが残念です。

・iPhone を用いた複数 PC の協調操作システム開発

これは iPhone を PC のマウスポインタの操作に使えるようにするシステムです。それだけではなく、ファイルを iPhone と PC 間で移動させることができます。まだ制作途中で、今後複数の PC を一つのデスクトップとして利用出来る機能などを追加する予定とのこと。実際に iPhone の画面をぐるぐるさせたりしてみましたが、中々やってて楽しいです。iPhone ユーザーはもっと使いやすく感じるのかなーと思いました。

終わりに

いかがだったでしょうか？ ごく一部しか紹介出来ませんでしたが、他にも便利なシステムや、面白いシステムの紹介がありました。この記事を読んで興味を持たれた方、将来、自分が大学院に進んだ時にこのプロジェクトに参加しているかもしれない、と思った方などは、来年行ってみてはいかがでしょうか。筆者はとても勉強になりましたし、大学院や教授の方々と色々なお話を出来て、有意義な時間を過ごすことが出来ました。

文明の進歩の尺度への道(1/4 dt)

文 編集部 Flast

はじめに

前号では CUDA の概要について簡単に説明しました。今回から 4 回の予定で実際に CUDA を使った応用プログラミングを行ってみようと思います。

早速今回は連立一次方程式を解いてみようと思います。1 年生も（私も 1 年生ですが…）線形代数 I で触れたと思うのでちょうどいい内容だと思います。

解き方には書き出し法を使います。本来なら LU 分解の方が面白いかもしれません、筆者が勉強不足ということもあり、うまい並列化の方法を思いつくことができなかつたので、ゴリ押しで書ける書き出し法にしました。もし LU 分解やコレスキーフィルタが上手くできたら是非筆者にご一報を！

前号の簡単なおさらいと応用の為の予備知識

前号までの知識では CUDA プログラミングをするには不足しすぎているため、おさらいも兼ねて新しいことをいくつか紹介します。まずおさらいから。

- CUDA では GPU 側を Device、CPU 側を Host と呼ぶ。
- Device 側のメモリを確保するには malloc() や new 演算子ではなく cudaMalloc() を使う。
- メモリのデータを転送するには memcpy() ではなく cudaMemcpy() を使う。
- メモリを解放する時は free() や delete 演算子ではなく cudaFree() を使う。
- スレッドの集合を Thread Block と、更に Thread Block の集合を Grid と呼ぶ。
- 新しい構文として、関数が実行されるデバイスを指定する __device__ と __global__。
- Device 側の関数を実行するための三重山括弧 <<< >>>。
- スレッドを識別する為の変数 threadIdx。

前号の簡単なおさらいとしてはこんな所でしょうか。しかし応用プログラミングをするには、まだ CUDA に関する知識が必要になります。具体的には以下のようないくつかの制限があります。

- Thread Block でのスレッドは xyz の方向にそれぞれ最大 512, 512, 64 分割。
- Thread Block 内合計スレッド分割数の最大は 512。
- Grid 内での Thread Block を識別する為に必要な変数 blockDim、blockIdx。
- Compute Capability というハードウェアのバージョンがあり、1.3 以降でないと倍精度浮動小数点(double 型)が使えない。
- スレッドが実行される順序は不定であり、どのスレッドから実行されるかは実行時にならないと分からぬ。

1 つ目と 2 つ目に関しては簡単です。つまり 512x1x1 や 128x2x2 のような分割方法は許されるが、512x2x2 や 768x1x1 のような両方の条件を同時に満たさないような分割方法は許されないということです。

3 つ目については threadIdx と同じような扱いですが、大量にスレッドを生成する時（具体的には 512 を超える数を生成するとき）には必要になります。具体的な使用法はソースを交えながら解説します。

文明の進歩の尺度への道(1/4 dt)

4つ目の Compute Capability ですが、2009年11月現在までに1.0, 1.1, 1.2, 1.3の4種類があります。

このうち1.3以降でないとdouble型を使用することができません。それ以前のGPUで小数を扱う場合、float型を使うしかありません。とりあえず本記事ではCompute Capability 1.0を対象にしています。1.0を対象にしたソースコードはCUDAの動く環境下すべてで動くためです。

ちなみに1.3に対応したGPUは最新モデルのみとなっています。GeForceでいえばGT200シリーズにあたります。どちらにせよ最新モデルのみで使用可能な機能なのでピンポイントで対象にしない限り使用しないでしょう。

5つ目に関してはマルチスレッドプログラミングを行ったことのある人でも分からぬかもしれません。つまりCUDAでは同期をとることがMutexやCriticalSection^{*1}を使うように簡単に行えません。Thread Block内の同期はとれますか、Thread Blockを越えた同期はとることができません。そのため全てのスレッドが参照する元データを変更する場合は注意が必要です。

さあ、これらの知識を元にCUDAプログラミングをしましょう…と言う前に環境の構築をしましょう。

環境構築

CUDAでのプログラミングを行うには、Toolkitという専用のコンパイラや動的リンクライブラリが必要になります。また、実機で試すにはドライバも必要になります。これらCUDAのコンパイラやドライバ、実行する為に必要な動的リンクライブラリは、CUDA Zone^{*2}からダウンロード^{*3}することができます。執筆時点での最新バージョンは2.3です。

CUDAはWindows、Linux^{*4}、Macに対応していますが、筆者はMacを使用していないのでこれに関する記述は省かせていただきます。

Toolkitのインストールが完了したらnvccをインストールしたディレクトリにパスを通しておくと便利でしょう。また、nvccは内部でC/C++コンパイラを呼び出します。

Windowsの場合はMicrosoft Visual C++ 2005/2008のcl.exeが必要になります。Express Editionsが無償で提供されているのでそれをダウンロード^{*5}してインストールすれば良いでしょう。

Linuxの場合はgccが必要です。それぞれのディストリビューションの方法でインストールしてください。

これらコンパイラもパスを通しておく必要があります。パスを通さない場合nvccの引数に-ccbinを指定する必要があります。また、インクルードパス/ライブラリパスも-I-Lで指定する必要があります、出力ファイルは-oで指定できます。追加のオブジェクトファイルはコマンドの最後に追加していくべきリンクが何とかしてくれます。

CUDAはDeviceで実行されるコードをHost上で動作するようなコードを生成し、エミュレーションすることができます。試したい場合はコンパイラオプションに-deviceemuを追加するとHost

*1 Windows独自のスレッド同期の仕組み。

*2 http://www.nvidia.co.jp/object/cuda_home_jp.html

*3 http://www.nvidia.co.jp/object/cuda_get_jp.html

*4 対応しているディストリビューションは必ずしも多いわけではありませんが、今後の更新で増えていくでしょう。

*5 <http://www.microsoft.com/japan/msdn/vstudio/express/>

上で動作するようなコードを生成します。もちろん GPU による超並列の恩恵を受けることはできません。が、試してみるだけならエミュレーションでも問題ない^{*6}でしょう。

パスを全て通した状態で `hoge.cu` というソースから `hoge.out` というバイナリを生成し、`obj1.o` と `obj2.o` というオブジェクトファイルとリンクするには以下のコマンドで実行できます。

```
nvcc -o hoge.out hoge.cu obj1.o obj2.o
```

また、エミュレーションするには

```
nvcc -deviceemu -o hoge.out hoge.cu obj1.o obj2.o
```

となります。

細かいオプションは-hで見ることができます。

まずは前号のソースがコンパイルでき、実行できる事を確認してください。実行には Windows では `cudart.dll` が、Linux では `libcudart.so.2` が必要です。

Windows 上で実行しようとしたとき `cudart.dll` が見つからないと怒られた場合、パスに `cudart.dll` があるディレクトリを追加してください。デフォルトのインストールディレクトリは `C:\CUDA\bin` または `C:\CUDA\bin64` です。

Linux 上で実行しようとしたとき `libcudart.so.2` が見つからないと怒られた場合、`/etc/ld.so.conf` に `libcudart.so.2` があるディレクトリを追加して `/sbin/ldconfig` を実行してください。デフォルトのディレクトリでは `/usr/local/cuda/lib` です。

連立一次方程式を解く

さて、ここからは本格的に実装していきましょう。

まず前提条件として今回扱う方程式では、全ての解が一意に求まるものとします。また、n 元連立一次方程式を解くには n 個の式が必要ということなので、拡大係数行列は $n \times n+1$ 行列となります。

方針としては掃き出しのみを Device 上で実行するようにします。また、1 スレッドに 1 方程式を割り当てて実行します。

全体を載せると長くなるので、いくつかの関数に分けながら説明を行います。それぞれがやっていることを理解するのは難しいかもしれません、頑張ってひも解いていってください。

関数の説明の前にいくつか必要なヘッダファイルのインクルードと定義します。

```
#include <float.h>

const int MAXIMIZE_SIZE_OF_THREADBLOCK = 512;

inline void SafeFree( float *&ptr )
```

^{*6} ただし Device で実行される場合と同数のスレッドを Host 側で生成するので、スレッド数が多いとオーバーヘッドが大きくなり、非常に時間がかかります。

文明の進歩の尺度への道(1/4 dt)

```
{  
    if ( ptr )  
    {  
        free( ptr );  
        ptr = NULL;  
    }  
}  
inline void SafeCUDAFree( float *&ptr )  
{  
    if ( ptr )  
    {  
        cudaFree( ptr );  
        ptr = NULL;  
    }  
}
```

MAXIMIZE_SIZE_OF_THREADBLOCK は Thread Block の最大サイズを示します。SafeFree() と SafeCUDAFree() はメモリ解放のときによく使われる関数です。

これらの定義を元に関数ごとに分解してソースを見ていきましょう。

まず最初の関数は GetColVector() です。これは Device 側のメモリ上にある拡大係数行列の特定列を取得するものです。

```
bool GetColVector( float *d_mat, float *h_col, unsigned long height, unsigned long col )  
{  
    unsigned long width = height + 1;  
    if ( !d_mat || !h_col )  
    {  
        return false;  
    }  
  
    return cudaMemcpy2D( h_col, sizeof( float ), d_mat + col,  
                        sizeof( float ) * width, sizeof( float ), height,  
                        cudaMemcpyDeviceToHost ) == cudaMemcpySuccess;  
}
```

最初の数行は単に引数のチェックです。本題は cudaMemcpy2D() です。

これは cudaMemcpy() の亜種で、単純ではないコピーを行うために使います。特にこの cudaMemcpy2D() は矩形にメモリをコピーすることができます。

第 1 引数はコピー先のアドレス(dest)、第 2 引数はコピー先のピッチ(dpitch)、第 3 引数はコピー元のアドレス(src)、第 4 引数はコピー元のピッチ(spitch)、第 5 引数はコピーする幅(width)です。これら 3 つの引数はバイト単位で指定する必要があります。

ピッチというのが最大幅で第 5 引数の幅というのは取り出す幅です。第 6 引数は高さ(height)、第 7 引数はどのメモリからどのメモリへコピーするかのフラグです。

この関数の挙動は非常に分かり辛いので、慣れるまでいろいろと試してみると良いかもしれません。行列で表すと、

$$\left(\begin{array}{ccc} SRC_{0,0} & \cdots & SRC_{0,width-1} \\ \vdots & \vdots & \vdots \\ SRC_{height-1,0} & \cdots & SRC_{height-1,width-1} \end{array} \right)$$

という行列の左側 $height \times width$ の矩形部分を

$$\left(\begin{array}{ccc} dest_{0,0} & \cdots & dest_{0,width-1} \\ \vdots & \vdots & \vdots \\ dest_{height-1,0} & \cdots & dest_{height-1,width-1} \end{array} \right)$$

とコピーします（結局分かり辛いですね）。

`GetColVector()` ではこの `width` を 1 要素分にしていると言うことです。とりあえずこの使用法だと列ベクトルを取り出すことができると覚えてください。

次に掃き出し法で必要になる行の入れ替えを行う関数 `SwapRow()` です。

```
bool SwapRow( float *d_mat, unsigned long width, unsigned long i, unsigned long j )
{
    if ( i != j && !d_mat )
    {
        return false;
    }

    float *d_tmp = NULL;
    bool ret = cudaMalloc( ( void ** )&d_tmp, sizeof( float ) * width ) == cudaSuccess
        && cudaMemcpy( d_tmp, d_mat + ( i * width ), sizeof( float ) * width, cudaMemcpyDeviceToDevice ) == cudaSuccess
        && cudaMemcpy( d_mat + ( i * width ), d_mat + ( j * width ), sizeof( float ) * width, cudaMemcpyDeviceToDevice ) == cudaSuccess
        && cudaMemcpy( d_mat + ( j * width ), d_tmp, sizeof( float ) * width, cudaMemcpyDeviceToDevice ) == cudaSuccess;

    SafeCUDAFree( d_tmp );
    return ret;
}
```

`cudaMemcpy()` の第 4 引数に `cudaMemcpyDeviceToDevice` を指定すると VRAM 内でコピーを行います。毎回 Host にコピーする必要がなく、効率的に転送が行えます。

&&演算子を多用してカオスなことになってますが、結局やってることは一時領域を確保して `i` 行目と `j` 行目を入れ替えています。&&演算子を使っているのでどこかの関数が失敗すれば、つまり `cudaSuccess` 以外を返した場合 `ret` に `false` が代入され、関数自体が `false` を返すようになっています。

別にこの様にする必要は無いですが、`if` のネストとかが嫌いな筆者はこの様な方法をとっています。

次は掃き出しのときに使われるピボットと呼ばれる係数行列の要素を含む行を取得する関数

文明の進歩の尺度への道(1/4 dt)

GetPivot() です。

掃き出し法では n 回目の掃き出しに使われる $a_{n,n}$ 要素を **ピボット** と言います。このピボットが 0 になると零除算が発生してしまうため、演算を続けられなくなってしまいます。また 0 でなくとも 0 に限りなく近ければ除算をした際に誤差が大きくなってしまうため、このピボットの**絶対値が最大になるように**方程式を選んで入れ替えます。これを"部分ピボット選択"と言い、先ほどの SwapRow() が使われます。部分と言うからには"完全ピボット選択"というのもありますが、こちらは演算量が非常に多くなってしまうので今回は行いません。

とりあえずソースを見てください。

```
bool GetPivot( float *d_mat, float *d_pivot, unsigned long height, unsigned long col )
{
    if ( height <= col || !d_mat || !d_pivot )
    {
        return false;
    }

    float *h_col = ( float * )malloc( sizeof( float ) * height );
    if ( !GetColVector( d_mat, h_col, height, col ) )
    {
        return false;
    }

    bool ret = false;
    if ( fabs( h_col[ col ] ) < FLT_EPSILON )
    {
        int max_col = col;
        float max     = fabs( h_col[ max_col ] );
        for ( int i = col + 1; i < height; ++i )
        {
            if ( max < fabs( h_col[ i ] ) )
            {
                max_col = i;
                max     = fabs( h_col[ max_col ] );
            }
        }
        ret = max_col != col && SwapRow( d_mat, height + 1, col, max_col );
    }
    else
    {
        ret = true;
    }
    SafeFree( h_col );
    return ret && cudaMemcpy( d_pivot, d_mat + ( height + 1 ) * col,
                           sizeof( float ) * ( height + 1 ),
                           cudaMemcpyDeviceToDevice ) == cudaSuccess;
}
```

第 4 引数の col はどの行・列にある要素がピボットになるかを表します。なのでその col 列目を取得してその col 番目の要素が"**0 に限りなく近い**"場合は行の入れ替えを行っています。

この"**0 に限りなく近い**"という判定には float.h で定義されている **FLT_EPSILON** 定数を使っています。この定数は float 型で表せる 1 より大きい最小値と、1 との差を表していてよく収束判定等に用いられます。

今回はピボットの絶対値と比較して、FLT_EPSILONの方が大きかった場合は0に限りなく近いと判定^{*7}しています。もしdouble型を使う場合は同じような定数でDBL_EPSILONが定義されているのでそちらを使うといいでしよう。

FLT_EPSILONとの比較で"0に限りなく近い"と判定された場合は、絶対値が最大の式を探してその式に入れ替えていきます。入れ替える対象となる式はまだピボットになったことのない式です。つまりcol行目からheight-1行目までの間にある式です。

最後にピボットがある行だけ別領域にコピーしています。これは予備知識のところで説明した通りスレッドの実行順序が不定のため、どのタイミングで変更されるか分からない値は別領域にコピーしておく必要があるからです。

本来ならDevice上で動作するkernel^{*8}を先に定義しておくべきですが、kernelを呼び出す側を作ってしまいましょう。

それほど難しいことはしていないくて、今まで作ってきたものをうまくつなげていくだけです。

```
bool Sweep( float *h_mat, float *ans, unsigned long height )
{
    unsigned long width = height + 1;
    bool ret = false;
    float *d_mat = NULL, *d_pivot = NULL;
    if ( cudaMalloc( ( void ** )&d_mat, sizeof( float ) * height * width ) == cudaSuccess
        && cudaMemcpy( d_mat, h_mat, sizeof( float ) * height * width,
                      cudaMemcpyHostToDevice ) == cudaSuccess
        && cudaMalloc( ( void ** )&d_pivot, sizeof( float ) * width ) == cudaSuccess )
    {
        ret = true;
        for ( int i = 0; i < height; ++i )
        {
            if ( !GetPivot( d_mat, d_pivot, height, i ) )
            {
                ret = false;
                break;
            }

            dim3 dimBlock = dim3( MAXIMIZE_SIZE_OF_THREADBLOCK );
            dim3 dimGrid = dim3( ( height + dimBlock.x - 1 ) / dimBlock.x );
            sweep_kernel<<< dimGrid, dimBlock >>>( d_mat, d_pivot,
                                                       make_ulong2( width, height ), i );
        }
    }

    ret && ans && GetColVector( d_mat, ans, height, height );
}
SafeCUDAFree( d_mat );
SafeCUDAFree( d_pivot );

return ret;
}
```

*7 実際近いことは確かですが数学的な限りなく近いではなく、演算によっては桁落ちや情報落ちしてしまうほどの近さということです。

*8 __global__で指定された関数はしばしばkernelと呼ばれます。

文明の進歩の尺度への道(1/4 dt)

まず Host 側で用意した拡大係数行列を VRAM へ転送し、ピボット行用の領域を確保しています。そしてループ内では元の数だけ掃き出しを行います。sweep()は実際に掃き出しを行う kernel です。

Thread Block や Grid の分割を指定する dim3 型の変数は、初期化のときに y,z 要素を省略すると **1で初期化**されます。特に dimGrid の方を注目してもらいたいのですが、この様な形で初期化するとたとえ元の数が MAXIMIZE_SIZE_OF_THREADBLOCK を超えたとしても総計スレッド数が元の数以上になるような Grid に分割します。つまり元が MAXIMIZE_SIZE_OF_THREADBLOCK より 1 でも大きければ Grid を 2つ以上に分割し、512 スレッドを内包した Thread Block を 2つ以上生成します。

ここで注意ですが、Grid は dim3 型の変数を用いるのにもかかわらず **2次元までの分割**しかできません。Grid の z 要素は常に 1 になるようにしてください。

こうしてスレッドは Thread Block により、Thread Block は Grid により、うまく割り当てられます。もちろん元以上のスレッド数になる可能性があるので、スレッドの制御は kernel 内で行います。

全ての係数に対して掃き出しが行えた場合、拡大係数行列の最右列を取り出すとこれが求める答えとなります。ピボットの選択で用いていた GetColVector()がここでも使えます。

大体の処理の流れはこれまで定義してきた関数を眺めると分かるかと思います。では、実際に掃き出しを行う sweep_kernel()を見てみましょう。

```
__global__ void sweep_kernel( float *d_mat, float *d_pivot,
                             ulong2 size, unsigned long col )
{
    unsigned long row = blockIdx.x * blockDim.x + threadIdx.x;
    if ( size.y <= row )
    {
        return;
    }

    float a = d_mat[ row * size.x + col ];
    if ( fabs( a ) < FLT_EPSILON )
    {
        return;
    }
    float pivot = d_pivot[ col ];
    for ( unsigned long cur = col; cur < size.x; ++cur )
    {
        float &res = d_mat[ row * size.x + cur ];
        if ( row == col )
        {
            res /= pivot;
        }
        else
        {
            res -= d_pivot[ cur ] * ( a / pivot );
        }
    }
}
```

まず変数 row に注目してください。sweep()の呼び出し時に Grid でも分割を行ったので blockIdx と blockDim という新しい変数を使っています。Thread Block 内ではスレッドを 512 に分割してい

るので threadIdx.x の値の範囲は 0 から 511 までです。512 を超えたスレッドを扱うときは Grid も使うので、実行しているスレッドが内包されている Thread Block が Grid の中で何番目の Thread Block かを識別する為に blockIdx を使います。

Grid も xy 方向に分割することができますが、今回は x 方向だけなので blockIdx.x のみ使用します。blockIdx.x の値の範囲は 0 から dimGrid.x-1 までです。

これだけだと各スレッドと行を一意に対応付けることができません。そこで使うのが、Thread Block 内のスレッドの個数を記憶している blockDim 変数です。ここでは Thread Block 内でスレッドが 512 あることになるので blockDim.x は 512 です。この値は全てのスレッドで同じ値なので、ソースの様に blockIdx と掛け、threadIdx を足すと一意に割り当てることができます。

あらかじめ行列のサイズは引数で渡しているので y 要素、つまり行数とスレッドの row を比較し行列の範囲外になったスレッドはその時点で終了する様にしています。これを行わないで範囲外のスレッドも動作させると、確保した領域外のメモリを破壊しながら実行されるので非常に危険です。もったいないと思われるかもしれないですが、これが CUDA プログラミングの定石です。覚えておくようにしましょう。

次に、書き出しが行われる列に該当する係数を a として記憶しておきます。もしこの値が 0 に限りなく近い場合、書き出しを行う必要がないことを示します。また、現在選択している行がピボットを含んでいる場合、書き出し法の性質上、他の行と同じように演算できません。そのため条件分岐を行い正しい演算を行っています。

これら関数を使ってランダムに生成した拡大係数行列を解いてみようと思います。

```
#include <time.h>
#include <stdio.h>

const int GEN = 100;

int main( void )
{
    srand( time( NULL ) );
    float *ans = ( float * )malloc( sizeof( float ) * GEN );
    for ( int i = 0; i < GEN; ++i )
    {
        ans[ i ] = ( ( rand() % GEN ) - ( GEN / 2 ) );
    }

    float *CoeMat = ( float * )malloc( sizeof( float ) * GEN * ( GEN + 1 ) );
    for ( int row = 0; row < GEN; ++row )
    {
        for ( int col = 0; col < GEN; ++col )
        {
            CoeMat[ row * ( GEN + 1 ) + col ] = ( rand() % GEN ) - ( GEN / 2 );
        }
        float b = 0.0f;
        for ( int i = 0; i < GEN; ++i )
        {
            b += CoeMat[ row * ( GEN + 1 ) + i ] * ans[ i ];
        }
        CoeMat[ row * ( GEN + 1 ) + GEN ] = b;
    }

    float *proc_ans = ( float * )malloc( sizeof( float ) * GEN );
    Sweep( CoeMat, proc_ans, GEN );
}
```

文明の進歩の尺度への道(1/4 dt)

```
    SafeFree( ans );
    SafeFree( proc_ans );
    SafeFree( CoeMat );

    return 0;
}
```

GEN は方程式の元の数を表します。とりあえず今回は 100 にしましたが、RAM と VRAM が許す限り増やすことができます。

解と拡大係数行列の生成には乱数を用いました。Sweep()での演算結果は proc_ans に格納されます。ans との比較をとってみると面白いかもしれません。ただし書き出し法は演算量が多く、また CUDA では float 型しか使えないでの、元の数が増えると誤差も大きくなります。

ベンチマーク

実際に私が実機で試した結果をリストアップしてみようと思います。

CPU で同等の処理を行った場合と、GPU で演算した場合の 2 通り。元の数を 10, 50, 100, 500, 1000, 2000, 3000 と増やしてみて実行しました。GPU でのコードは Thread Block 内のスレッド数による実行速度の変化を見るために 128, 196, 256, 512 と増やしてみました。また、CPU での等価なコードは大体同じような感じなので割愛します。

計測に使ったのは clock() です。これは実行にかかった CPU 時間を取得することができる関数で、計測対象の前後で時間を取りすればそれらの差が実行にかかった時間を示すようになります。

元	CPU	GPU / 128	GPU / 196	GPU / 256	GPU / 512
10	0.000	0.031	0.031	0.031	0.031
50	0.000	0.031	0.031	0.031	0.031
100	0.000	0.031	0.031	0.031	0.046
500	0.500	0.328	0.297	0.328	0.468
1000	3.875	2.641	2.485	2.625	2.766
5000	572.359	517.031	526.266	514.125	533.907

clock() が返す時間の単位は秒ではありませんが、下の表では秒換算しています。

また、今回使用した CPU は Intel Core2Quad Q6600 で、GPU は NVIDIA GeForce 9600GT です。Host 用のソースはシングルスレッドで実行しました。

残念なことに CPU に比べ GPU ではあまり高速化できませんでした。いくつか理由をあげることができます。

- CUDA ではメモリの使い方が重要になり、多くのスレッドが同時にアクセスしようとすると処理能力が落ちる。
- そもそもアルゴリズムがよくない。
- 筆者が悪い。

などです。下の 2 つはこの際おいておいて、1 つ目のメモリについて少し説明したいと思います。

Device のメモリについて

種類	最大サイズ	スコープ	キャッシュ
Local Memory	16KB/Thread	Thread	x
Global Memory	VRAM の限界まで	Global	x
Shared Memory	16KB/MultiProcessor	Thread Block	x
Constant Memory	64KB	Global	o
Texture Memory	テクスチャのサイズによる	Global	o

CUDA で使えるメモリにはいくつか種類があります。以下に列挙します。

これら 5 種類のメモリをプログラマは使うことができます。このうち前回と今回の記事で使っているのは上の 2 つです。特に説明しませんでしたが、上の 2 つに関しては C 言語などにおけるローカル変数やグローバル変数にあたります。また、`cudaMemcpy()` で Host 側から Device 側へコピーしたときに使われるのも Global Memory です。

これらのメモリはキャッシュされないので、何度もアクセスする様なプログラムを書いた場合動作は遅くなります。しかしキャッシュされないメモリの 1 つである Shared Memory は、条件が揃えばレジスタ^{*}並みの速度でアクセスすることができます。

とはいって、Shared Memory や Constant Memory、Texture Memory は上手く使うにはテクニックが必要となる為、今回は説明しません。とりあえずそのようなメモリがあることぐらいは覚えておいてください。いつかは説明したいと思っています。

今回のまとめ

簡単に今回の要点だけをまとめてみたいと思います。まず

- ・ 1 つの Thread Block では最大 512 スレッドが実行される。
- ・ Grid を使うと blockDim 変数や blockIdx 変数によってより多くのスレッドを識別できる。
- ・ 実は倍精度浮動小数点(double 型)が使えない。
- ・ `cudaMemcpy2D` という関数はメモリを矩形に切り取ってコピーする。
- ・ やたらとメモリの種類がある。

といったところでしょう。

本来なら最適化とかまでやりたかったのですが、紙面の都合上とりあえず簡単にわかる程度にしました。最適化をしてみたい人は頑張ってみてください。多分掃き出し法の最適化をするより LU 分解を実装した方が高速かつ正確になると思いますが…。

*9GPU のコアに最も近く、アクセス速度は最速。ただし、プログラマはこれを直接いじることはできない。

文明の進歩の尺度への道(1/4 dt)

次回^{*10} の予告ですが、複素関数論の授業で Mandelbrot 集合を描くプログラムを作らされるらしいということを小耳にはさんだので、これをやってみたいと思います。期待しないで待っていてください。

*10 あくまで次回です。次号に載るとは限らないので注意してください。多分無理です。

GRな日々。

文 編集部 葡萄酒

GRD2との出会い

2年間、レンズを通して私と同じ景色を眺め続けた愛機” RICOH Caplio GX-8(以下GX8)”が死んだ。今年の春のことである。もともと中古で購入した上に湿気やら極寒やら過酷な環境で戦い続けてきたのだから、限界が来ても仕方ない時期であった。フォトコン、運動会、青少年美術展といろいろな場面で活躍してくれたGX8、今までご苦労様。本当に楽しかったよ。

ということで、私の手元にはカメラが無くなった。私は携帯電話を持たない主義であるため、全く写真が撮れない状況である。大学では授業やら何やらが忙しくなり、シャッターを切っている暇もないだろうから問題あるまいと踏んでいたのだが——ぶっちゃけ暇である。加えて、私にとって「写真が撮れない」という状況は想像以上に苦痛であった。

さんざん悩んだ末、財布をはたいて新しい相棒を購入することにした。機種は、前から憧れを抱いていた” RICOH GR Digital II(以下GRD2)”に決定。下に軽くスペックを掲載しておこう。

- ・撮影素子:有効画素数 1001万画素、1/1.75型原色CCD
- ・レンズ :焦点距離 5.9mm (35mm換算値 28mm)
F値 F2.4～F11 (オート撮影モード時 F7.1以上は、NDフィルター併用)
撮影距離範囲 レンズ先端から 約30cm～∞
レンズ先端から 約1.5cm～∞ (マクロ撮影時)
- ・レンズ構成 5群6枚 (非球面レンズ3面2枚)
- ・シャッタースピード:180、120、60、30、15、8、4、2、1～1/2000秒
(RICOHの製品紹介ページ^{*1}より抜粋)

このカメラは1996年にRICOHが発売した高画質コンパクトカメラ” RICOH GR1(以下GR1)”シリーズのデジタル版として2005年に発売された” RICOH GR Digital”の後継機である。銀塩のGR1からのウリであったGRレンズを搭載、独特の鮮やかな色彩とレンズ周辺の歪曲の少なさにより一眼レフにも勝るとも劣らない表現力を持つ。また、GRシリーズ黎明期より28mm単焦点レンズ^{*2}というスタイルを貫き続けてきた、コンパクトカメラとしてはある意味で異端とも言える硬派なカメラでもある。その玄人好みの性能から根強いファンも多く、プロの写真家にも一眼レフのサイドアームとして愛用している人が少なくない。

などと、GRの良さについて語り始めるとキリが無いので、この辺で一言にまとめると——

「男のロマン」 だよ、うん。これに勝る言葉は見あたらない。

*1 <http://www.ricoh.co.jp/dc/gr/digital2/spec.html>

*2 焦点距離を変更できない、つまり光学ズームの使えないレンズを指す。通常のズームレンズに対して軽くて明るい(F値を小さくしやすい)上に安価であり、同コストでは高性能なものが作れる。ただし、撮影者の技術とセンスが問われるため、ファミリー向けが主な客層であるコンパクターマニアでの搭載機種は非常に珍しい。

白神山地旅行記

夏休みの中頃に、編集部 一七夜月氏から旅行のお呼びがかかる。目的地は青森県、白神山地である。今思えばこの旅行に参加してよかったと心から思える。その理由については、これから挙げる写真を見れば納得していただけるだろう。

週末の ETC 割引を駆使して高速に乗り、数時間ごとに運転を交代しながら^{*3} 東北に向かう。余談ではあるが、私は 7 月に免許を取ったばかりである。そして旅行に行ったのが 8/23。まともに高道路を走ったことがない中での運転は、スリル溢れるエキサイティングな体験であった。

旅行自体は 2 泊 3 日の予定ではあるものの目的地までの距離があるので、一日目と三日目は移動のみで、実際の観光もとい撮影は 2 日目のみの予定であった。しかし、初日は予想以上に早く宿に到着してしまい、近場に小さなダムがあると聞いて早速そちらに向かった。ダム自体は非常に小規模なものであり取り立てて面白いものではなかったが、タイミング悪く（あるいは良く）降り出した雨によって、実際に幻想的な風景を見ることができた。



RICOH GR Digital II

ISO80 / 1/100sec.

F9 / 0EV / WB:Auto

雨上がりの空に虹。紙面ではわかりづらいが、濃い 1 本の外側に淡く 2 本目の虹が見えるのがお分かりいただけるだろうか。

また、山の緑、空の青と湖面に浮かぶ赤とのコントラストが美しい。

さて、本命の 2 日目である。ヘアピン続きの細い山道を越えて、目的地へと向かう。やはり下界と比べて少し肌寒く、しかも天候が怪しい。目的の滝へは車が入れず、川沿いに山道を登っていかなければならない。今にも降り出しそうな空だったので雨具を購入し、名所「暗門の滝」へ。

滝へと続く道に足を踏み入れると、そこは別世界だった。木々は青く、流れる川はどこまでも透明。せせらぎの音、草木のざわめき、虫の声が一体となって何とも快いハーモニーを奏で、それはさながら大自然のオーケストラのようであった。不安定だった天候もすぐに回復し、雲の隙間から差し込む太陽の光が渓流の水面に反射して、きらきらと輝いている。

*3 同行したのは私と一七夜月氏に加えて yasuharu 氏、suma 氏、いのひろ氏の計 5 名。実は私が高速を運転したのは帰路だけだったりする。

RICOH GR Digital II
ISO80 / 1/40sec.
F9 / 0EV / WB:Auto

川辺に佇む、羽化したばかりの蝉。いわゆる蝉の茶色ではなく、赤みがかった橙色の筋に、驚くほど透明な羽。まだ飛ぶことができないのだろう、レンズを向けても逃げることなく落ち着いていた。



更に道は険しさを増してゆく。岩肌に作られた不安定な足場、急な登り道が続く。こちらが立ち止まっていると、帰ってきた人とすれ違えないため、呑気に写真を撮っている暇もない。鉄パイプの足場を頼りに第三の滝、第二の滝^{*4}を超えて、最奥にある第一の滝へ。

進むこと数十分。岩場に囲まれていた視界が急にひらけ、待ち望んでいた第一の滝が姿を現す。力強い音と、飛沫を上げて飛び散る水流。その自然の力がもたらす迫力は筆舌に尽くしがたいものであった。



RICOH GR Digital II
ISO80 / 1/10sec.
F9 / 0EV / WB:Auto

奥に写っているのが暗門第一の滝。落差は 42m である。木々が生い茂る中、隙間を縫って流れ落ちる滝はどこか神秘的なものを感じさせる。激しい水音を迸らせながらも、何故か周り一帯は不思議な静謐さに満ちていた。

ここまで綴ってきた白神山地旅行記であるが、お楽しみ頂けただろうか。余談だが、実際の旅行では、美しい夕暮れの空に歓声を上げたり、”大富豪”による死闘が繰り広げられたり、帰りの車でかづきお氏がダウンしたりと色々あったが、ここでは触れないでおこう。

何はともあれ、この旅行は文句なく楽しかった。被写体に飢えていた私に、強いインスピレーションを吹き込んでくれた。イギリスの政治家、ベンジャミン・ディズレーリは「旅は真正な知識の偉大な泉である」と述べている。自分の知らない場所へ旅に出ると、常に新しい発見を得ることができる——なんと素晴らしいことだろうか。故に、この記事が読者の皆様の旅へと踏み出すきっかけとなれば幸いである。もちろん、旅に出るときにはカメラをお忘れ無く。

*4 3つある滝は、上流側から順に”第一の滝”、”第二の滝” というように名前がつけられている。

入門 Ruby on Rails Vol. 4

文 編集部 いのひろ

おしらせ

Ubuntu のアップデート (9.10) で Rails のバージョンが 2.2.3 に更新されますが、特に問題ないと思われます。本記事では Ubuntu 9.10 (Rails 2.2.3) を使っていきます。

本題

さて、前回は実際に Model や Controller を編集して Rails アプリケーションに変更を加えてみました。データ入力の時により簡単に項目を選択できるようにする変更でした。

New article

Title

Author

Category

Book

Start page

前回の変更を加えたあとの articles のトップページは以下のようになっていると思います。

Title	Author	Start page	End page	Category	File	Book
article_1	0	10	0	Show Edit Destroy		
article_2	1	30	1	0	1	Show Edit Destroy

New article

しかし、このままだと Author、Category、Book の列が数字で表示されているので、なにがなんだか良くわかりません。ということで、前回はデータ登録のところをいじってみましたが、今回はデータ表示のところをいじってみようと思います。

表示したい情報（Author であれば編集者の名前、Category であればそのカテゴリ名、Book であれば冊子タイトル）を牽引するための情報（ここでは Article.author_id や Article.category_id, Article.book_id）はそろっているので、これで find^{*1} してあげればよいわけです。

app/views/articles/index.html.erb を開きます。14 行目から始まる for 文の中を変更します。下のコードで「<%#」から始まる行はコメントアウトされて実行されます。今回はあらかじめ書いてあったコードを比較のためにコメントアウトして残しています。必要なければ消してしまって大丈夫です。

```
<% for article in @articles %> <%# @article は Controller の index メソッド内で定義されている
<tr>
  <td><%=h article.title %></td>
  <td><%=h Author.find_by_id( article.author_id ).name %></td>
<%#     <td><%=h article.author_id %></td>
  <td><%=h article.start_page %></td>
  <td><%=h article.end_page %></td>
  <td><%=h Category.find_by_id( article.category_id ).name %></td>
<%#     <td><%=h article.category_id %></td>
  <td><%=h article.file_id %></td>
  <td><%= Book.find_by_id( article.book_id ).title %>
<%#     <td><%=h article.book_id %></td>
  <td><%= link_to 'Show', article %></td>
  <td><%= link_to 'Edit', edit_article_path(article) %></td>
  <td><%= link_to 'Destroy', article, :confirm => 'Are you sure?', :method =>¥
:delete %></td>
</tr>
<% end %>
```

app/views/articles/index.html.erb

*1 データベースから情報を引っ張ってくるメソッド

入門 Ruby on Rails Vol.4

この for 文の中では、app/controller/articles_controller.rb の 5 行目で定義してある@articles の中身を列挙しています。@articles は RDB の Article テーブルのすべてのデータが格納されています。script/console で確認してみましょう。

\$ script/console

```
Loading development environment (Rails 2.2.3)
```

```
>> Article.find( :all )
```

```
=> [#<Article id: 1, title: "ror_vol4", start_page: 0, end_page: 20, category_id: 1, file_id: nil, book_id: 1, created_at: "2009-11-10 07:14:28", updated_at: "2009-11-10 07:14:28", author_id: 1>, #<Article id: 2, title: "ror_vol4_2_2", start_page: 1, end_page: 30, category_id: 1, file_id: nil, book_id: 1, created_at: "2009-11-10 07:17:29", updated_at: "2009-11-10 07:20:25", author_id: 1>]
```

script/console でデータなどを列挙するときには「pp」を使うと多少見やすくなります。pp は「pretty-print」のことと、このライブラリを使うと表示を整形してくれます。

pp を使ってみましょう。pp を使うには、事前に「require 'pp'」する必要があります。

```
>> require 'pp'
```

```
=> ["PP"]
```

```
>> pp Article.find( :all )
```

```
[#<Article id: 1, title: "ror_vol4", start_page: 0, end_page: 20, category_id: 1, file_id: nil, book_id: 1, created_at: "2009-11-10 07:14:28", updated_at: "2009-11-10 07:14:28", author_id: 1>, #<Article id: 2, title: "ror_vol4_2_2", start_page: 1, end_page: 30, category_id: 1, file_id: nil, book_id: 1, created_at: "2009-11-10 07:17:29", updated_at: "2009-11-10 07:20:25", author_id: 1>]
```

```
=> nil
```

ちょっと分かりにくい知れませんが、レコードごとに改行してくれているのがわかります。

ということで、@articles には script/console で確認したように、article のレコードデータが入っています。「for article in @articles」はこれを列挙しています（article はループ内変数です）。

17 行目 「<td><%=h article.author_id %></td>」を見てください。列挙されるデータは table タグの中の一つの要素として HTML に変換されます。そのため「<td>」タグで囲まれているのは納得できると思います。その中の「<%=」以下ですが、すぐに「h」があります。この「h」、実はメソッドで「html_escape」の別名です^{*2}。

*2 Ruby はメソッド呼び出しの括弧を省略することができます。「h(article.autho_id)」と書いても同じです。

`h (html_escape)` メソッドは何をやってくれているのでしょうか。また `script/console` で調べてみましょう。

\$ script/console

```
Loading development environment (Rails 2.2.3)
>> h 'hello'
=> "hello"
```

特になんの変化も起こっていません。では、たとえばこれが HTML タグだったりすると、

```
>> h '<h1>Hello, World</h1>'
=> "&lt;h1&gt;Hello, World&lt;/h1&gt;"
```

このようにエスケープしてくれます。RDB は「`<script>`」タグなどは文字列として扱いますが、これを Rails がそのまま HTML に埋め込むと何かよくわからないスクリプトを実行されてしまう恐れがあります。これを防ぐために `h (html_escape)` メソッドを使うようにしましょう。

さて本題に戻ります。`/articles` を表示したときに `Author` の列が数字で表示されてしまうのは、`article.autho_id` の中身が数字だからです。このデータをもとに `Author` の名前(name) を取得してみましょう。先ほどの一行を

```
<td><%= h Author.find_by_id( article.author_id ).name %></td>
```

に書き換えます。サーバーを起動して `http://localhost:3000/articles` にアクセスします。

\$ script/server

`Author` のところが数字ではなく人の名前になっているのが確認できましたか？ 同じように `Category` と `Book` のところも変更してみましょう。

```
<td><%=h Category.find_by_id( article.category_id ) %></td>
```

入門 Ruby on Rails Vol.4

```
<td><%=h Book.find_by_id( article.book_id ) %></td>
```

Title	Author	Start page	End page	Category	File	Book
ror_vol4	Hiroyuki Inoue	0	20	Tech.	word_12	Show Edit Destroy
ror_vol4_2_2	Hiroyuki Inoue	1	30	Tech.	word_12	Show Edit Destroy
<H1>HELLO, EORLID</H1> Hiroyuki Inoue						
Tech.						
word_12 Show Edit Destroy						

New article

articles/index と同じように articles/show/id のときにデータの詳細が表示されますが、これも app/views/article/show.html.erb を変更することで、数字ではなく名前を表示することができます。

もしかしたら確認の過程で「NoMethodError in Articles#index (もしくは Article#show)」に遭遇した方がいるかもしれません。

Action Controller:

You have a nil object when you didn't expect it!
The error occurred while evaluating nil.name

Extracted source (around line #23):

```
20:  
21: <p>  
22:   <b>Category:</b>  
23:   <%=h Category.find_by_id(@article.category_id).name %>  
24: </p>  
25:  
26: <p>
```

RAILS_ROOT: /home/inohiro/Projects/word-30th

エラーの詳細には、

```
You have a nil object when you didn't expect it!  
The error occurred while evaluating nil.name
```

と書いてあつたりするはずです。これは「Author.find_by_id(@article.category_id).name」などをしたときに、find した結果が nil である（該当するレコードが無い）にもかかわらずその name プロパティにアクセスしてしまった為、例外が発生してしまったのです³。

とりあえずこのエラーに対応するには、出力するところが「find した結果が nil (null) であれば、"NULL"を出力する。そうでなければ find した結果の name プロパティを出力する」というロジックにすればよいと考えられます。

```
<% if ( cat = Category.find_by_id( @article.category_id ) ).nil? %>
<%=h "NULL" %>
<% else %>
<%=h cat.name %>
<% end %>
```

このコードでは Category.find した結果を変数 cat に代入、それが nil であれば"NULL"と出力、そうでなければ cat の name プロパティを出力ということをしています。



しかしこれでは対症療法になってしまいます。本来は登録時に入力された値をチェックすべきです。次回は Web フォームに対する Validation (検証) をかける方法から始めたいと思います⁴。

*3 Java でのぬるぼ (java.lang.NullPointerException) です。ガッ

*4 予定では Validation まで今回書く予定でしたが、collection_select を用いた View では validation_presence_of (フォームの項目が空っぽでないことを検証するメソッド) がうまく動かないという現象に遭遇したので、今回はここまでにしておきます。Controller で params[:title].blank? とする方法もありますが、データに関するロジック (とくに入力データの検証など) は Model に書くのが望ましいので最適とは言えません。

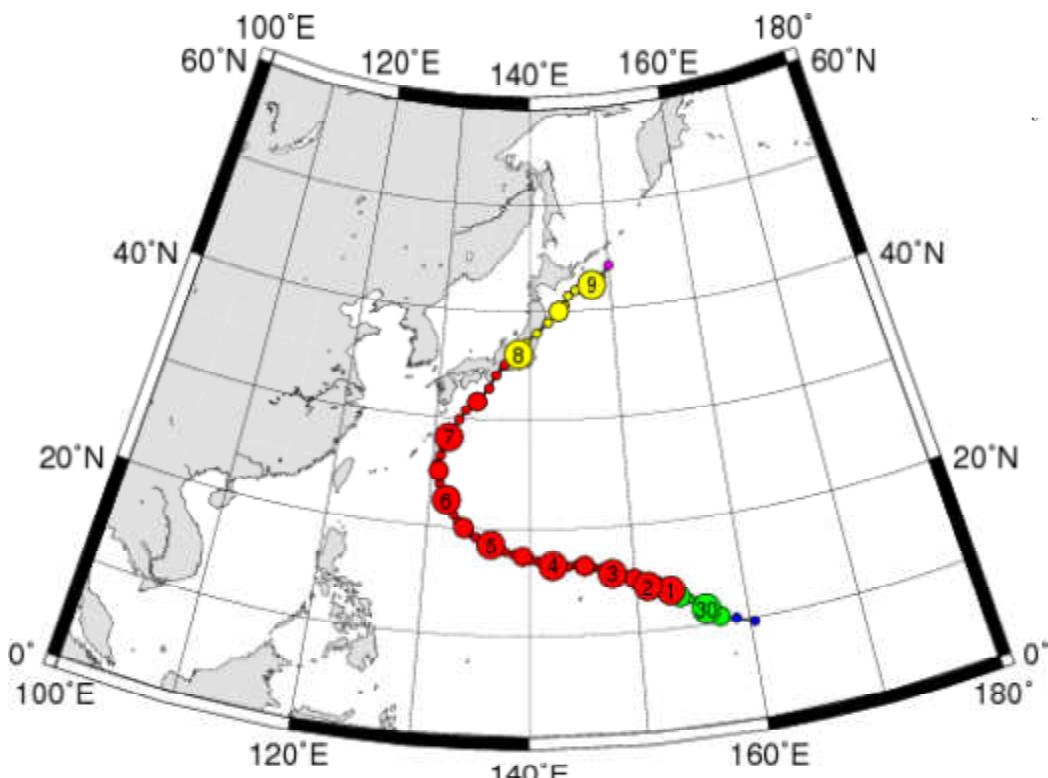
台風と戯れる Zwei!

文 編集部（一七夜月）

最強クラスの台風、現る！

超大型！ 伊勢湾台風と同じ進路！ 関東直撃！ などと騒がれた台風を諸君は覚えているだろうか。その名はメーローたん^{*1} こと台風 18 号である。

その大きさときたらなんとハンバーガーが 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 個ぐらいかな。



発生日時	2009-09-29 12:00:00 UTC	消滅日時	2009-10-09 06:00:00 UTC
継続期間	234 (時間) / 9.75 (日)	最低気圧	910 (hPa)
最大風速	110 (knots)	最大強風域	480 (km)
最大暴風域	1710 (km)	移動距離	6348 (km)
平均速度	27.1 (km/h) 651 (km/d)	移動幅	緯度33.4 (度) : 経度26.9 (度)

すごく……、おおきいです……。

*1 メーロー (Melor) : 台風 200918 号 (正式名称) のアジア名

よろしい、ならばアクアラインだ

編集部員 K 「L 先輩、台風ですよ！ Typhoon！」

編集部員 V 「アクアライン!!!1!^{*2}」

大型台風にはしゃぐ編集部員たち。休講の知らせも届き浮き足立つ。

編集部員 L 「でもあの車^{*3} じゃ心細いんだよねー。パワーないしな。」

メーローたんのあまりな強大さに弱気になる L。しかし、次の瞬間均一なるマトリクスの裂け目の向こうから

編集部員 G 「実家のワゴン車だせるかも……。」

ざわ…ざわ…

ざわ…ざわ…

というやりとりがあったとかなかったとか。とりあえずアクアラインへ向かう。



g しかし、そんな一同をあざ笑うかのような雨、風。センターインの見えない常磐道を飛ばし、視界の悪さで道を間違えながらたどり着いた編集部員の見たものは果たして……

*2 アクアライン： 台風が来たらアクアラインへ向かうのは WORD 編集部の伝統なのである。

*3 あの車： 編集部員 L の車。豊田紡織製。

台風と戯れろ

「あれ、雨降ってなくね？」

メーローたんと戯れる

まあそんな心配は杞憂に終わったわけで、すぐに雨も風も強くなってきた。よしよし。



暴風雨の中でもアクアラインの美しさは健在。

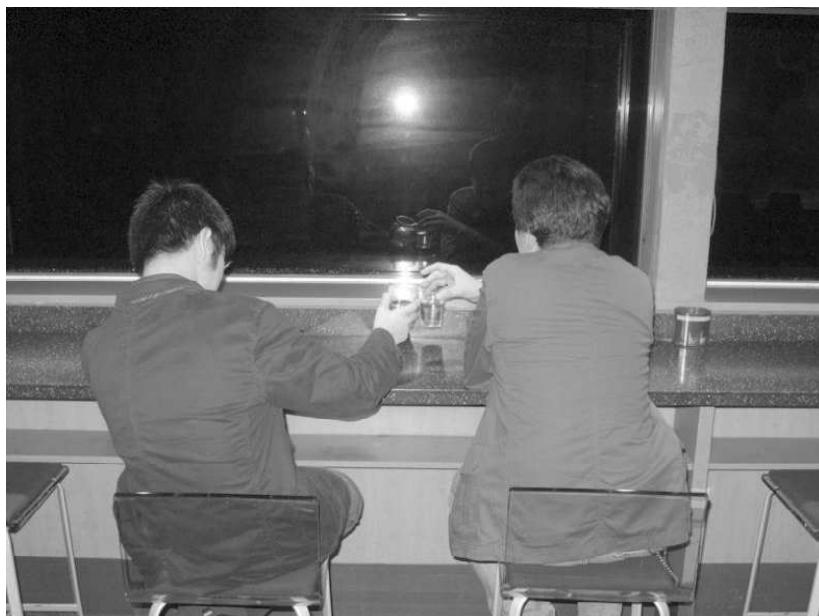
最接近時まで軽く腹ごしらえをしながら時間をつぶした編集員は午前 5 時頃再び上甲板に昇った。

到着後よりも雨は弱まっているようだが、風は強まることはあっても弱まる気配を見せない。それでもメーローたんと戯れ続けた我々に彼女が残してくれたのは鉄屑となった 4 本の傘であった。



そして午前 6 時、我々はメーローたんに別れを告げ帰宅の途についたのだった。

メーローたんに



乾杯！

長編大作 「男の娘と箱ガンダムと」

文 編集部 dorio

生け贋 編集部 yasuhiro, inohiro, Santarh

取材 編集部一七夜月, なるがみ, 葡萄酒, Flast

某月某日、編集部。

学園祭一日目昼下がり。学園祭の音をバックサウンドに、僕たち編集部員は各自の作業に没頭していた。そんな平和な時間を、ある編集部員の声が断ち切った。

「なあ、彼女居る？」

ざわめきが広がる。おい空気読めよだとか、お前のホームディレクトリ吹っ飛ばすぞ、だとかいう声が聞こえてくる。

「居たらこんな日に学校来てる訳無いだろ。」憮然とした顔で、一人の編集部員が答えた。「お前、本当に空気読めないな。空気の読めない WORD 編集部員の中でも、とびきりだ。俺が保証してやる。」

その時、また別の編集部員の声が響いた。

「俺たちは大学で何を学んだんだ? **無い物は作ればよい**、そう、その高貴なる物作りの精神じゃないか。」

衝撃が走った。そうか、という思いと共に可能なのだろうか? という疑問も同時に浮かんだ。

「それは現実的な解か? 実現可能性が低い物に投資するほど僕たちは暇じゃあない。」

別の編集部員がそう言い、部内を見回した。その後にしかし、と言葉を次いだ。

「でも、面白い。そういう新しいことをする為に、僕たちはここにいる。僕たちがやらねば誰がやる。**僕たちが通った後が道なんだ。**」

学園祭に行くには、何が足りないだろうか。

学園祭に必要なものは彼女以外にも色々ある。しかし、僕たちの中にも、その知り合いにもリア充が居ない為、その必要なものの洗い出しが困難を極めた。そこで出された案は次のようなものだった。

「似たものを分析し、そこから類推する。」

これは様々な分野の学問で用いられている、非常にオーソドックスな手法である。ではどんな物が学園祭に似ているだろうか。なおかつ、それが僕たちが良く知るものではないといけない。そこから自ずと導きだされる物は、そう、**コミケ**である。

コミケ、それは世界一の祭典。思い浮かぶのは同人誌、人ごみ、コスプレ、そしてカメコ^{*1} で

*1 カメラで男の娘を写すことが好きな人。 カメラ小僧の略。

ある。同人誌と人ごみは元々クリアしているし、幸い WORD にはカメコも居る。しかし、問題はコスプレだ。そう、**人前に着ていく服が無い。**

彼女と服。これをどうやって確保するか。僕たちは長く熱い議論を交わした。「女性をどこから連れて来る」「その女性にコスプレをして貰えば良い」そのような案が出たが、協力してくれるような女性なんて、知り合いに居るはずもない。しかし、ここでも DIY(Do It Yourself)精神が遺憾なく発揮された。

「男の娘を作れば、コスプレも彼女も同時に条件を満たせる」

そう、僕たちは人前に着てゆく服はなかったが、**女 装 服**なら持っていた。

女性を DIY



どうだろう、女性も嫉妬を覚える程の男の娘ではないだろうか。

と、ここで新たな編集部員が現れた。その**華奢な身体に定評がある**、yasuharu であった。「あれ、皆さん何やっているんですか？」

華奢な身体、そして中性的な顔立ち。そう、**男の娘にピッタリな逸材**であったのだ。「君、かわいいね。いいよ、ちょっとこれを着てみないか？」編集部員一同 yasuharu にじり寄り、そして服を剥いでゆく。「ちょ、ちょっとやめて下さい……！」しかし、彼の声は誰かの耳に届くことはなく、一輪の花はポトリと落ちてしまったのだ。

長編大作: 男の娘と箱ガンダムと



編集部内にため息が漏れる程の出来映えだった。ああ、こんなにも可愛い娘が女の子であるはずがない！その言葉の意味を再認識した瞬間であった。

しかし、ここで大きな問題が発生した。女装する服が足りない。
せいそうふく 女装 服を持っているのが半ば常識と化して久しい情報科学類ではあるが、その普及率は 100%には届かない。どうしても、女装用の服が足り無くなってしまったのだった。何か服を DIY する材料は無いものか。編集部を捜索し、その材料を集めたところ以下のものが見つかった。

- ・メモリ (SD-RAM)
- ・猫のぬいぐるみ(非常に可愛らしい)
- ・蛇腹のホースのようなもの
- ・紙コップ
- ・CPUx2(Athlon thunderbird, Intel Pentium!!!)
- ・段ボール
- ・キーボード (Dell)

せいそうふく
これでは女装 服を作るのは難しそうだと、編集部は暗い雰囲気に包まれた。しかし、この時ある編集部員が段ボールを見て聞いた。「段ボールと言えばガンダムだ。」

ガンダムを DIY

段ボールと言えばガンダム、ガンダムと言えば段ボールとまで言われる程に一般常識になった箱ガンダム^{*2}。制作に手間が掛からない割に、その機能性は高い。残念ながら女装着とは言えないものの、イベントの場に相応しいインパクトのあるものだし、作った女性をはべらせる役としても相応しい。そんな高貴な役に、僕 dorio/m-bird が選ばれたのであった。

*2 箱ガンダム: 別名「段ボールガンダム」。"GUNDAM"とマジックで書いてある段ボールを着込んだ外国人男性の写真は、余りにも有名だろう。



通常の箱ガンダムを、WORD の総力を結集してチューンナップした、特製「箱ガンダム Dual CPU Edition」。WORD の技術力は伊達じやない。そうして、我々は満を持して学園祭へと殴り込みに行つたのである。

学園祭を歩く

学園祭中のキャンパスの雑踏の中を、箱ガンダムと麗しき男の娘たちが歩いてゆく。「ちょっと風が吹くとパンツが見えそうになりますね。」「女性用下着を準備すべきだったかもしれないね。」そんな他愛の無い会話をしながら体芸棟に向かう。目指すは「芸バー」である。

芸バーは、筑波大学学園祭で最も有名なものの一つで、芸術専門学群(略してゲイ専)の一年生男子たちが女装をして接待してくれるという、非常に素晴らしい出物である。WORD 編集部員の一団も「一度は行ってみたい」と思っていたものの、リア充たちの中を歩いて行くという危険を冒せるはずもなく、結局噂で「マジ凄いらしい」というのを聞くだけであつ

た。

しかし、今年は僕たちは男の娘+箱ガンダム。リア充共なぞ怖くもない。そして、もし芸バーの中にモノホンが居たとしても、**女装をしていれば怖くない**。箱ガンダムも居るし、万に一つ手を出されそうになつても蹴散らしてくれるわ！と高笑いしながら芸バーに向かうのであつた。

芸バーへ

改装され、美しくなった体芸中央棟。そこに、箱ガンダムと麗しき男の娘たち。目指す芸バーは5階、エレベータは混んでいた。「しかたない、階段を使おう」そう言って、僕らは階段に足を掛けたところで、カメコの一七夜月はある重大な事実に気づき、そして叫んだ。

「君らは今男の娘だ！スカートの中を覗かれてしまう！恥じらいを持て！エレベータを使うんだ！」

長編大作: 男の娘と箱ガンダムと

僕らはハッとなった。そう、格好だけでは「男の娘」には成り得ないのだ。気持ちが、理解が、情熱が足りなかった。僕たちは自分たちの至らなさを恥じ、反省しつつエレベータ待ちの列に並んで芸バーへと向かった。



芸バーに着くと、そこには案内役の女性が立っており、取材の旨を説明すると「なるほど、分かりました。大丈夫です。ゲイとの時間を楽しんで下さい」と、快諾してくれた。さすが、筑波学園祭に名だたる芸バーである。

しばし待ち、部屋に通されるとそこは大学の一室とは思えないほど「バー」を再現していた。薄暗い部屋の中、女装したゲイたちが訪れた客たちの接待をしていた。そして、その女装の完璧さに僕たちは圧倒され、立ち尽くした。

すると、ほどなくして僕らの席にゲイの人がやってきた。「あらやだ、あなたたち凄いわね！」しかし、彼は非常に美しい脛、そして腕をしており、その暗闇の中でもそのきめ細やかな手入れの行き届いた腕に、編集部員一同思わず息を呑むほどであった。



食べながら歓談したり、じやれ合つたり。また、「ゲイがしているプラジャーを誰が一番早く外せるか」競争などを眺めたりしているうちに、あっという間に規定時間である 20 分を越えてしまった。僕たちは、後ろ髪を引かれる思いで芸バーを後にすることになった。



長編大作: 男の娘と箱ガンダムと

学園祭を楽しむ

第一の目的、芸バー行きを果たした僕たちは、次に"そおしあーる"を目指した。"そおしあーる"とは、WORDと双璧を成す学類誌であり、常に熾烈なシェア争いが繰り広げられている。先号ではセンタープレを宣戦布告されて先手を打たれてしまったが、こちらにも反撃する準備は幾つもある。まずはお互い交流をしながら笑顔で腹の探り合いをするべきだ。そうして、そおしあーるの焼き鳥ブースへと足を運んだ。

「いやちよ、どんな格好してるんですか w」「いや、WORDですから。」「えええ、凄い準備していたんですねえ……。」「いや急にやろう！って話題になったもので。いつでも女^{せいそう}装する準備をしておくのが紳士というものです。」

そんな会話を繰り広げつつ、美味しい焼き鳥を頂いて後にした。

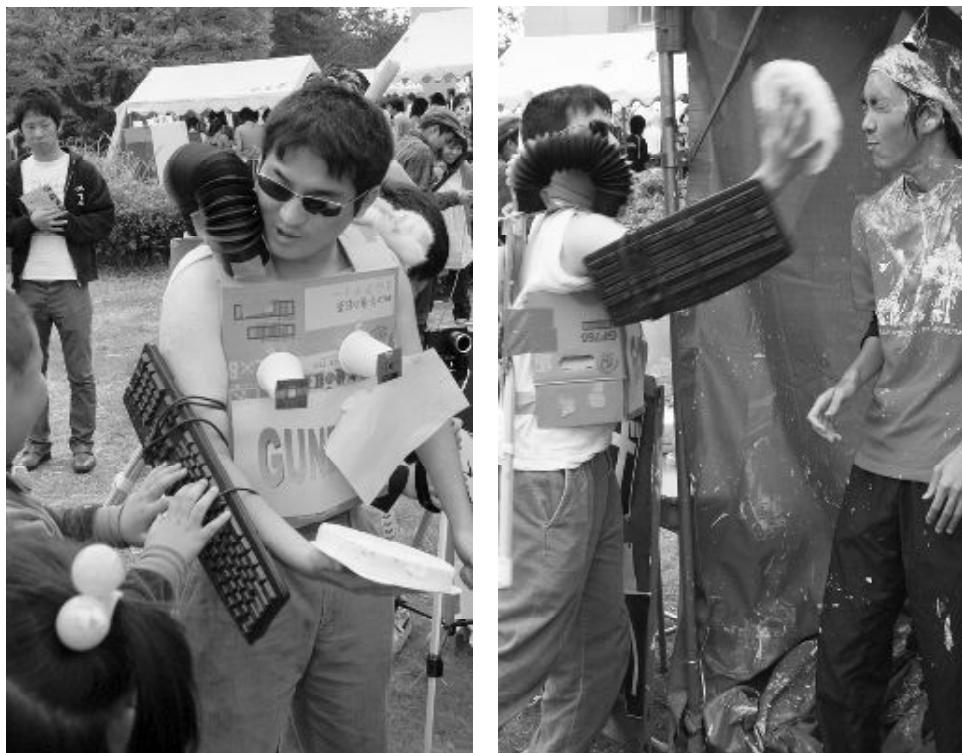


さらに構内を歩き回っていると、「パイ投げ無料ですよ！」「ああ、そのガンダムさん、是非パイを投げていって下さい！」と声をかけられる。一体何事だろうと見えてみると、生クリームまみれの服を着たイケメンが数人立っていた。「あの中からご指名して、是非パイをぶつけてやって下さい。」とパイを渡される。

無料ほど高いものは無い。しかし、ガンダムがやらねば誰がやる！と僕は右手のパイをそのままイケメンの顔にぶつけた。バジュリ、というパイの潰れる音が響き、ずぬるりとパイが落ちると、生クリームまみれの(元)イケメンフェイスが現れた。

本当にこれは無料なのだろうか？後から肝臓だとか要求されないだろうか、とビクビクしていると、「ガンダムさんありがとうございましたー」とそのまま会場を後にすることとなった。

パイ投げの慈善事業。そんなものは聞いたことがない。しかし、このストレスが溜まる現代社会では、それを発散する手段として色々必要なのだろう、と思いを巡らせながらその場を後にした。



宴の終わり

華奢な体型の二人組(yasuharu,Santa)は非常に可愛らしく、各方面から好評で、各方面から「アンコール！ アンコール！」「もっともっと！」と熱狂的なラブコールが届いているが、yasuharuは黙したままである。一方、Santaは非常にノリノリで、きっと彼は将来有望な選手に育ってくれるだらうことは想像に難くない。

また、青いカツラにワンピースの inohiro は、可愛いというよりも大人びた男の娘になれたようで、歩いている時に屋台から「ちょっとそこのグラマスな女性、是非一個買って行ってよ！」などと声を掛けられることが何度もあった。彼は「**グラマス男の娘**」という新しいジャンルを切り開くべく、活動の幅を広げてくれることを願いたい。

そして、箱ガンダム—— dorio/m-bird は非常に複雑な思いを抱いているところである。というのも、**某 Campus ○ JT 部屋**で酷く話題になっているそうだからだ。「まじヤベえし www」 「俺みちやったし www 流石にあれはないわ www」と盛り上がっているとか盛り上がりたいとか。僕はそんな噂を耳にしながら、箱ガンダムに目覚める訳でもなく、今は至って普通の生活に戻っている。

長編大作: 男の娘と箱ガンダムと

最後に

一時の気の迷い、それがまた人生を左右することも確かだ。女装や仮装というのは果たして、本当に僕らに何も残さなかったのだろうか。もしかしたら、この企画はWORDの編集部員の人生を大きく変えてしまう程の大イベントだったのかもしれない^{*3}。

今日もまた、日本のどこかでAmazon箱に納められた女^{せいそうふく}装^{ふく}服^{ふく}が発送されている。



綺麗な格好してるだろ……？これ、男の娘なんだぜ？

超変態作: 「男の娘と箱ガンダムと」 (完)

*3 少なくとも、xxx がそちらの方面に興味を抱きつつあるのは確かだ。

糸編集後記

(文字: 編集部 ふあい)

情報科学類誌

WORD

From College of Information Science

WORDは一太郎を使っています号

発行者

情報科学類長

編集長

中 裕太郎

製作・編集

筑波大学情報学群
情報科学類 WORD 編集部
(第三エリア C 棟 212 号室)

2009 年11月 初版第一刷発行